Linguistic Spaces:
Kernel-based models of natural language

von

Armin W. Buch

Dissertation
zur Erlangung des akademischen Grades
Doktor der Philosophie
in der Philosophischen Fakultät
der Eberhard Karls Universität Tübingen

Tübingen
(2011)

Gedruckt mit Genehmigung der Philosophischen Fakultät der Universität Tübingen
Hauptberichterstatter: Prof. Dr. Gerhard Jäger
Mitberichterstatter: Prof. Dr. Fritz Hamm
Dekan: Prof. Dr. Jürgen Leonhardt

Abstract

This thesis discusses ways to employ a certain mathematical characterization of *similarity*, *kernel functions*, and machine learning techniques building on it to abstract from data-oriented models of language.

A prominent task in machine learning and in linguistic modeling is the *classification* of data items, the recognition of grammatical features. A bit more complex is the modeling of surface-to-surface relations, e.g. in inflectional paradigms. These morphological changes reflect the underlying grammatical features.

Traditional rule-based approaches to feature recognition and inflection fail to explain local regularities within the so-called exceptions. Many linguistic domains exhibit *islands of reliability* (Albright 2002) both within classes of exceptions and the regular cases. For its best-studied instance, the debate on the proper treatment of this phenomenon is known as the *past tense debate* (see Pinker and Ullman 2002).

More recent models (along with the increase in available computational power) adapted data-orientation, first in the shape of *prototypes*, and later as full-fledged *exemplar models*. These model *grammar* as the collective episodic memory of a speaker's experience with language. New utterances are classified or generated in *proportional analogy* (de Saussure 1916) to previous ones: if two pairs of linguistic items instantiate the same grammatical relation, the proportion of the first pair is analogous to that of the second.

Analogy crucially relies on a notion of *similarity*. Similar linguistic items are more likely to be classified alike. Learning paradigmatic relations via proportional analogy goes beyond learning as classification (Pirrelli and Yvon 1999, Albright 2008). It extends to surface-level outputs.

Kernels are a mathematical formulation of similarity. Essentially, they are an inner product in a feature space, where each dimension is an observable feature of the data items. Kernel methods are closely related to exemplar models (Ashby and Alfonso-Reese 1995, Jäkel et al. 2008), and they are successful in cognitive modeling, mostly outside linguistics. In this thesis I devise methods to generate linguistic models using *Kernel Principal Component Analysis* (KPCA; Schölkopf and Smola 2002). I cover models of classification (predicting the *gender* of German nouns) and of production (predicting the *plural* form).

Models of production require a model's output, which is a representation in the *continuous* feature space, to be mapped onto a *discrete* valid data item. I devise a general strategy of the required *pre-imaging* for complex data, which incrementally approximates the desired item, and instantiate this strategy for the data type of strings.

I compare KPCA to a variant including weights on data items, Weighted KPCA (Wang et al. 2005). I prove WKPCA's equivalence to a KPCA performed on a data set where weights are simulated via multiple identical entries of data items. In the experiments, WKPCA performs inferior to KPCA, probably due to over-generalization of the most frequent patterns.

Further, I investigate the linguistic interpretation of the *principal components*. These form the orthogonal basis of the feature space as calculated by KPCA. It turns out that — as is often the case with data-orientation — they do not represent abstract, scientific, human-readable descriptions of linguistic problems. I also discuss rotations of the principal components as alternative bases. I devise a method of rotation towards abstract features which retains all variation within these features in a low-dimensional model. By this I obtain a *two*-dimensional model of German gender, in which the three genders form clusters. While this is a practical model, it is still not an interpretable one.

Most models presented here are morphological in nature. In the syntactic domain, I extend the implementation of *grammaticality* as *planar languages* (Clark et al. 2006) from string languages to trees, using a *tree kernel* (Collins and Duffy 2001a,b).

Zusammenfassung

Diese Dissertation behandelt verschiedene Möglichkeiten, mittels einer bestimmten mathematischen Charakterisierung von *Ähnlichkeit*, *Kernel-Funktionen*, und darauf aufbauenden maschinellen Lernverfahren über datenorientierte Sprachmodelle zu abstrahieren.

Die *Klassifizierung* von Datenpunkten, ist eine der häufigsten Anwendungen maschinellen Lernens, und auch in der Linguistik findet sie als Erkennung grammatischer Eigenschaften Anwendung. Konzeptuell anspruchsvoller ist es, die Beziehung zwischen sprachlichen Ausdrücken zu modellieren. So realisieren zum Beispiel Wörter innerhalb eines Flexionsparadigmas die verschiedenen Merkmale, entlang derer flektiert wird.

Ausnahmen und gerade auch systematische Ausnahmen stellen sowohl in der Merkmalserkennung als auch in der Flexion die traditionellen regelbasierten Erklärungsansätze in Frage. Nicht nur innerhalb der Ausnahmen, sondern auch für die als einheitlich erachteten regelmäßigen Fälle, sind sogenannte *islands of reliability* (Albright 2002), kleine Gruppen ähnlicher Ausdrücke mit daraus folgenden ähnlichen Eigenschaften, beschrieben. Wegen der in der Literatur oft studierten Vergangenheitsform englischer Verben trägt die Debatte um die angemessene Behandlung dieses Phänomens den Namen *past tense debate* (siehe Pinker and Ullman 2002).

Spätere Modelle orientierten sich — angesichts steigender Rechenkraft — mehr und mehr an Daten, zuerst in Gestalt von *Prototypen*, und zuletzt als ausgewachsene *Exemplarmodelle*. Als *Grammatik* gilt darin das gesammelte episodische Gedächtnis des Sprechers für Sprache. Neue Äußerungen werden in *proportioneller Analogie* (de Saussure 1916) zu den alten verstanden beziehungsweise gebildet: Wenn zwei Paare sprachlicher Ausdrücke den selben grammatischen Bezug untereinander haben, dann ist auch ihr Bezug hinsichtlich ihrer Form ähnlich.

Analogie basiert auf einem Konzept der *Ähnlichkeit*. Ähnliche sprachliche Ausdrücke verhalten sich ähnlich, gehören eher den selben Klassen an und flektieren ähnlich. Paradigmatische Beziehungen zwischen Wörtern zu lernen geht über simple Klassifizierung hinaus (Pirrelli and Yvon 1999, Albright 2008), weil hier nicht abstrakte Merkmale gesucht sind, sondern ganze Wortformen.

Mit *Kernel-Funktionen* lässt sich Ähnlichkeit formalisieren. Im Wesentlichen handelt es sich bei ihnen um ein inneres Produkt in einem aus den beobachtbaren Merkmalen sprachlicher Ausdrücke aufgespannten Raum. Kernelmethoden stehen in einem engen Verhältnis zu Exemplarmodellen (Ashby and Alfonso-Reese 1995, Jäkel et al. 2008) und werden — meist außerhalb der Linguistik — erfolgreich zur kognitiven Modellierung eingesetzt. Deswegen entwickele ich in dieser Dissertation Methoden um mittels kernelisierter Hauptkomponentenanalyse (*Kernel Principal Component Analysis*, KPCA; Schölkopf and Smola 2002) Modelle von Sprache zu erstellen. Dabei decke ich sowohl Klassifizierungs- (am Beispiel des *Genus* im Deutschen) als auch Produktionsmodelle (am Beispiel des *Plurals*) ab.

In Produktionsmodellen muss das jeweilige Ergebnis, ein Punkt im *stetigen* Merkmalsraum, zusätzlich auf einen *diskreten*, gültigen Ausdruck abgebildet werden. Zu diesem Zweck gebe ich eine

allgemeine *Pre-Imaging*-Strategie für komplexe Daten an, die den Zielausdruck schrittweise annähert. Ich exemplifiziere sie für Zeichenketten (*strings*, also Wörter).

Ich vergleiche die KPCA mit ihrer gewichteten Variante, *Weighted KPCA* (Wang et al. 2005). Ich zeige, dass die WKPCA zu einer KPCA über einen erweiterten Datensatz, in dem jeder Ausdruck gemäß seiner Häufigkeit mehrfach vorkommt, äquivalent ist. In den Anwendungen unterliegt die WKPCA jedoch der KPCA, was daran liegen mag, dass sie ohnehin schon häufige Muster übergeneralisiert.

Ferner untersuche ich die linguistische Interpretation der *Hauptkomponenten*. Diese sind die per KPCA berechnete, orthogonale Basis des Merkmalsraumes. Wie so oft in datenorientierten Ansätzen stellt sich heraus, dass sie keine abstrakt-wissenschaftlichen, menschenlesbaren Beschreibungen des jeweiligen linguistischen Problems sind. Ich erwäge Drehungen der Basis, um andere Interpretationen zu ermöglichen, und stelle eine Rotation auf die grammatischen Merkmale vor, die sämtliche Variation bezüglich dieser Merkmale in einem niedrigdimensionalen Modell darstellt. Dadurch erhalte ich ein beispielsweise *zwei*-dimensionales Modell, in welchem die drei Genera des Deutschen Cluster bilden. Obschon nützlich, ist diese Ebene immer noch nicht linguistisch interpretierbar.

Die meisten hier vorgestellten Modelle sind morphologischer Natur. Hinsichtlich der Syntax erweitere ich unter Verwendung einer Kernelfunktion über Bäume (Collins and Duffy 2001a,b) den Begriff der *Grammatikalität* im Sinne der *planaren Sprachen* (Clark et al. 2006) von Zeichenketten auf Baumdarstellung.

Acknowledgments

I wish to thank my supervisor Gerhard Jäger for giving me the opportunity to study an interesting and new field in linguistics, of which the present thesis can only cover an initial fraction. Its thematic focus was inspired by a talk given by R. Harald Baayen in Tübingen, December 2009.

This thesis owes its existence, its contents and its final form more people than just myself. I can and will take sole credit only for the remaining errors. The following people have contributed to this thesis with discussions, comments and as proof-readers (in alphabetical order; in hope I did not omit anyone): Anne Brock, Peter Buch, Kilian Evang, Cristophe Costa Florêncio, Magdalena Leshtanska, Christian Pietsch. I am most grateful to Johannes Dellert for proofreading the mathematical appendix.

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

This thesis is centered around the notion of *similarity*: its mathematical characterization on the one hand, and its role in analogical models of linguistic proportions on the other hand. It brings the state-of-the-art formalization of similarity in machine learning to data-oriented linguistics.

A measure of similarity is needed in any analogical reasoning and other types of cognitive modeling. A current trend in machine learning (ML) and artificial intelligence is based on similarity measures: so-called *kernel methods* (Schölkopf and Smola 2002). These ML methods are successfully applied to classification and recognition tasks. In this thesis, I also propose production tasks: kernel-based models outputting full word forms. Kernel methods are general enough to extend to any cognitive ability of humans which is based on association and similarity of episodic memories.

A prominent human cognitive ability is language. As a linguist, I like to point to language as the one ability that truly distinguishes our species from others. Its scientific description involves classification, recognition and production tasks. Applying kernel methods to linguistics suggests itself, for all their success in related fields. Also, as I will argue, they abstract over what is already being done in linguistics. There have been some initial proposals for doing so (discussed in section 2.3), but the general picture is still missing. For the ML practitioner, this thesis shall demonstrate some new ways of applying their successful and well-founded methods to the interesting domain of natural language.

The corresponding trend in linguistics is both very old and new. Old is the notion of analogy, going back at least to de Saussure (1916): If two linguistic items (words etc.) are similar, they behave in a similar way. That means, they are likely to have the same properties, to be used in similar contexts, or to be inflected analogically: The way they relate to the rest of the language will be similar.

This chapter sets up a wide range of linguistic classification problems (section 1.2). Classification is the easiest instance of a mapping from linguistic items into a target space. Following connectionism (Rumelhart and McClelland 1986, among others; see section 2.1.3), this will be modeled by analogy. If two words are similar on the *surface* (i.e., by their directly observable properties), then

they are likely to belong into the same class and to share abstract (grammatical[1]) features.

Considering not only two words, but a larger (training) data set, this translates into an input space where words are located according to their pairwise similarities. Test items not in the training set do not necessarily fall into the same space, since they may be independent of the training data. They can be projected onto that space, again according to their similarity. From their position within the data space they can inherit class labels and other abstract properties from their neighbors.

Similarity will be modeled very generally with a kernel function (section 2.2). The kernel is meant to capture all the psycholinguistic intuitions and insights about similarity. Kernel methods are closely related to exemplar models (Jäkel et al. 2009). The method to be used here is Kernel Principal Component Analysis (KPCA, Schölkopf and Smola 2002), and a variant sensitive to frequencies of data items, Weighted KPCA (Wang et al. 2005). It treats data items as points in a feature space, and derives an orthogonal basis for this space. In this space, analogical proportion can be modeled (section 2.4.3). I thus pursue a psycholinguistic approach with quantitative means.

In the light of modern machine learning, linguistics has often done explicitly what is done implicitly in a kernel: list all the properties of an item. It is the main goal of this thesis to show how linguistic models can be formulated using kernels — to show how the methodology that comes with kernels applies to linguistic data. The methods for non-deterministic, analogy-based classification and mapping tasks to be presented in this thesis are best suited for inflectional morphology, which I will concentrate on.

1.1.1 How to read this thesis

This thesis is primarily aimed at (computational) linguists, both with and without knowledge of kernel methods. I intend to provide a practical introduction to kernel methods for linguistics. The presented exemplary applications will be of limited interest to the general linguist; they are merely a proof of concept, although not bad in what they do. The methods can be applied to similar problems, also in other languages, but I have to leave the broader empirical evaluation to future research.

The appendix hosts all mathematical details, definitions, technical descriptions of data preparation and encoding issues, and algorithms. It is intended to serve as a manual for the methods developed here.

All URL's given in this thesis were accessible and had the described content in March/April 2011. Naturally, I cannot guarantee their future content or accessibility.

1.2 The task

This section outlines the range of linguistic tasks to be addressed in this thesis. All these tasks are *mappings* from some linguistic domain into another, the co-domain. In the simplest case, the co-domain is a fixed set of classes, or mutually exclusive *hidden features*. Most related work has been concerned with classifications.

[1] Abstract features in linguistics, at least the ones I will be considering in this thesis, are all *grammatical* features.

Analogical proportion (de Saussure 1916) views mappings as parallel (1.2.2): Similar items are mapped alike. This is captured in *exemplar models* (2.1.1). The section thereafter (2.1.3) discusses abstractions of those.

1.2.1 Classification

Classification is a popular task in machine learning and artificial intelligence. A set of objects is divided into pre-defined classes, as posited by the researcher. Class membership is a *hidden feature*, that is, it is not deterministic, but it may be deducible from the surface appearance of an object. Also in linguistics, classes and hidden (grammatical) features are widely in use.

Which features these are depends on the domain (e.g., words or sentences), on the language and on the concrete task. Examples are number, gender, person, case, aspect, tense and many more. They are posited by linguists, and they are needed to explain the ungrammaticality of utterances, where features are in conflict. That is to say, grammatical features can be observed indirectly, and they make sense as a parsimonious explanation of observable linguistic behavior.

Positing features is mainly a task for the researcher, although it is possible to learn feature geometries automatically. For now, I presuppose feature geometries for each given mapping task. Two linguistic tasks related to hidden features are relevant to this thesis: *extracting* (recognizing) and *changing* (inflecting for) features .

1.2.1.1 Extracting features

Given a surface form, the task is to determine the hidden features. In order for them to have an impact on grammaticality, they need to have some *exponent*, some overt marking. Otherwise they cannot be observed directly.[2] *Isolating* languages exhibit (by definition) a one-to-one mapping: Each word carries exactly one feature. Classification reduces to lexical look-up. In *agglutinative* languages, there is a separate morpheme for each grammatical feature; again this belongs to the characterization of this language type. Features can be extracted by parsing the word into morphemes, and looking up their meaning in the lexicon.

In *inflectional* (or *fusional*) languages, the situation is more interesting. Grammatical features may surface as internal changes to the stem. For example, umlaut is a marker of plural in German (cf. its remnants in English: foot — feet). However, plural is also expressed by other means (mostly affixes), and umlaut is an exponent of other grammatical features as well: It appears in the comparative and superlative of adjectives, in diminutives, and more.

So the relation between features and surface form is not a unique function in either direction. While there are deterministic morphological processes also in inflectional languages, there are these non-deterministic cases. Native speakers are able to master them, and on unseen words they agree at a level above chance, leaving the linguist at a loss to explain this. Chapter 3 is dedicated to this.

[2] A border case is the *null morpheme*, and overt, yet empty marker. Its existence follows from the feature geometry, that is, from its position in the given paradigm.

As a concrete example, take the gender of German nouns. Unless already known, deducing gender is a non-deterministic task. There are three classes (three possible values for gender), and every word falls into exactly one of them.

There appears to be a paradox: On the one hand, one tries to predict gender from surface forms. On the other hand, gender is informative beyond the surface form, e.g. in inflection (see section 1.2.1.2 below). The answer is that gender, base form, and inflected form are correlated. Gender of German nouns — to the dismay of language learners — is neither negligible nor redundant. Gender is an integral part of the lexical entry of a German noun. And still, the high correlation (which is also found among different native speakers in nonce word elicitation tasks) calls for a linguistic explanation.

1.2.1.2 Changing features

Keeping some properties of a word constant (such as the lemma), certain other features can be changed (such as number). Such changes to words are a key property of *inflectional* languages. This goes beyond classification (Pirrelli and Yvon 1999). In chapter 6 I build productive models of inflection.

A given word form has to be specified for all features appropriate to its part of speech. On the surface level, there can be no underspecification, yet there can be ambiguity. Some features are lexical, while for others a word can be inflected. An example of the former is gender of German nouns, an example of the latter is number. Note that this status depends on part of speech, and on other features: German adjectives do inflect for gender, but only if their number is singular. English distinguishes gender only with the pronouns (again, only in singular), and never inflects for it. For other parts of speech, such as verbs, gender is meaningless. Thus there are interdependencies between grammatical features.

As already mentioned above, there are several exponents of nominal plural in German. In fact, the choice of morpheme is non-deterministic, and still native speakers are able to extend their knowledge to unknown words, on whose inflection they agree more than could be predicted by either a random process or by a default rule. The special problem with the German plural is that the default case (suffixing -s) only covers 3% of the lexicon (see section 6.4.1). Such *minority defaults* challenge the idea of rule-based explanations.

Both feature extraction and inflection are mapping tasks. Extraction maps surface forms to abstract features, inflection maps surface forms to other surface forms (chapter 6). The linguistic intuition behind such mappings is that the hidden feature or inflected form can partially be predicted from what is known, although the process is in general non-deterministic. Gender is not directly marked at the surface level, yet it is predictable, at least to some degree (3.2.4).

1.2.1.3 Other classifications

The hidden features and abstract classes mentioned so far are mostly situated in *morphology*, because ignoring them results in ill-formed *words*. Classes are posited on other levels of linguistic analysis

as well.

In **phonology**, phonemes are described as feature bundles. There is, however, not a learning task involved: The set of phonemes needed to describe a certain language is always closed, so there are no unknown data items. In **phonetics**, where sounds are described not by abstract features but by physical measurements (vowel formants, durations, ...), classification is useful. Sounds could be classified feature by feature (each feature may take several mutually exclusive values), or as a whole (the pre-defined phonemes being the classes).

Among grammatical features, **morphological** ones determine inflectional forms: tense, number, aspect and so on. **Morpho-syntactic** features are not expressed by inflection, but inherent to a word, such as number for singularetanta, and gender for German nouns. Ungrammaticality then arises from *other words* having the wrong inflectional form, via (non-)agreement. Hence the term morpho-syntactic. Also, lemmatization (mapping inflected forms to a set of base forms or lemmas) and part-of-speech (POS) tagging (mapping words to a pre-defined set of POS labels) are mapping tasks.

Another type of inherent features are inflectional *classes*. These are used to describe lexically determined, different ways of expressing the same morphological feature. However, these classes differ from the usual hidden features. Gender has a fixed set of values for a given language, and the values are checked via agreement, thereby being observable. On the contrary, inflectional classes neither have to be a closed set nor do they stand in agreement to anything. They are merely descriptive, and not necessarily part of the grammatical description of a language.[3] A special instance of inflectional classes is the distinction of inflectional patterns into regular and irregular ones: The allophonic realizations of the English past tense (/-t/, /-d/, /-əd/) are considered regular, while everything else is labeled irregular.

The classification of words into inflectional classes can then be considered the non-deterministic pre-processing step to a deterministic second step: Once the class is known, the according rule realizes the surface form.

Classifications also exist in higher-level descriptions of language. **Syntactically**, sequences of words (or of POS labels) may form constituents and receive a phrase label (parsing). In a later chapter (7) I will come back to syntax. As a final example, yet not claiming exhaustivity at all, language recognition is a mapping, too: from texts to languages.

1.2.2 Analogical proportion

The domain can be considered unbounded: The generalization of a model has to be tested on previously unseen words. Typically, one investigates open word classes (content words) or other unbounded domains (sentences etc.). In order to generalize a model to unseen words, one has to look at input-output pairs in the context of other pairs instantiating the same relation.

[3] A side remark: Noun classes (as in Bantu languages) are much more like gender: A closed set of lexically determined features, therefore an indispensable grammatical property of a word. They do determine inflection, but they are not inflectional classes in the purely descriptive sense.

This is captured by the notion of *analogical proportion*. In modern linguistics, this goes at least back to de Saussure (1916). Albright (2008) gives the following definition:

(1.1) Four-part notation: A:B :: X:Y

"Whatever the relationship is between A and B, it should also hold between X and Y"

There are two basic types of relations between the different layers of annotation which are of interest here, identifiable by the ontological status of the co-domain. Assume that the domain always is a set of linguistic items. These are the real-world objects of linguistic investigation (or, rather, transcriptions thereof). Now the co-domain can either be of the same type. An instance of such a relation are word forms within an inflectional paradigm. The second type of the co-domain is abstract features posited by linguists. For instance, linguistic objects have grammatical properties, or fall into abstract classes (see 1.2.1).

For linguistic modeling, mapping tasks follow from this: First, from words to inflected forms. This is *production*. Second, from linguistic objects to abstract features. This is *classification*. There are more layers of annotation, more mapping tasks, but analogy-based models are most apt to describe these two.

Albright notes that the proportion extends from pairs to tuples. These encompass all layers of linguistic annotation to a linguistic object. For words, part of that is their inflectional paradigm. In a similar way, *paradigm-based proportional analogy* is defined in Pirrelli and Yvon (1999). They illustrate it like this:

(1.2) (drink, Present) : (drank, Past) :: (sink, Present) : (sank, Past)

Positions within the paradigm (that is, inflected forms) are identified by their grammatical features, here: tense. Not only pairs of paradigms stand in analogical proportion, *all* paradigms of verbal inflection do:

(1.3) (drink, Present) : (drank, Past) :: (sink, Present) : (sank, Past) ::[4] ... :: ...

A model trained on 1.3 could be presented with a task formulated as 1.4:

(1.4) (drink, Present) : (drank, Past) :: ... :: (stink, Present) : (X, Past)

Assume that "stink" has not been in the training set. It is treated according to the similarity of its present tense form to (present tense forms of) the training data. Now assume that past tense formation behaves analogically, thus the past tense of "stink" is similar to the past tense forms of words to which "stink" is similar in present tense. The question is how to formalize this notion. In the next section, I will discuss the network models of Rumelhart and McClelland (1986) and Plunkett et al. (Plunkett and Nakisa 1997, Plunkett and Juola 1999), and in chapter 6 I present my own model for this task.

[4]The relation of analogical proportion (::) is transitive and symmetrical.

1.2.2.1 Beyond classification

Linguistics is full of ever more complex tasks which could also be treated analogically. Assume that similar sentences have similar syntactic structures, similar meanings, or similar translations into another language. Analogical models for these tasks circumvent any rules, any grammar. They learn solely by example. In this thesis, I give only one example beyond morphology: Grammaticality is checked via similarity to other grammatical data items in section 7.2. The methods presented and introduced in this thesis can in principle be applied to all these tasks. They are of course limited at least by the upper bound of what analogy can actually explain about these linguistic processes.

1.2.3 Conclusion

Classification has been extensively studied in psychology and cognition (see Shepard 1987). The models tested in numerous experiments range from rule-based approaches via prototype models to fully distributed exemplar models and artificial neural networks, in that chronological order. Several factors have driven the development of new models. First, the notion of an *exception* made it necessary to extend purely rule-based models.

Second, sub-regularities within the exceptions have questioned the basic distinction between regular and irregular cases. A necessity for storing ever more prototypes was found, down to single exemplars. The impact of an exemplar on classification has often been modeled as dependent on the mutual similarity of the exemplar and the item to be classified: More similar stored items have a greater impact on the classification of the test item. This leads to clusters of similar items all belonging to the same class, so-called *islands of reliability* (Albright 2002).

Third, the increase in computational power made exemplar models feasible. Models always try to make the most of the available architectures, and so will the models presented here.

Classification has not only been studied in psychology, but also in linguistics as a field within cognition. There are linguistic classification and mapping tasks that are non-deterministic, yet learnable to some degree by native speakers. This calls for formal models and/or explanations. The basic insight is that new words are treated *analogically* to known words. The same topics apply here: the rule vs. exception debate (to be discussed in more detail as the *past tense debate* in chapter 6.1), prototype and exemplar models, effects of similarity, islands of reliability. Graded judgment studies have found graded grammaticality of regular and irregular inflections (see e.g. Albright and Hayes 2003).

Among linguistic classification tasks, the morphology of inflectional languages is most interesting to the methods that I discuss in this thesis. The general idea is to infer morphological and morpho-syntactic features of previously unseen words based on their similarity to known words. Similarity is surface-oriented, and it is calculated as an aggregate value over all observable features and parts of the words, as discussed in detail in section 2.2.

This relies on the hypothesis that surface and hidden features are not arbitrary, but correlated. The hypothesis is backed by elicitation studies, where native speakers achieve agreement above chance

on the hidden features of nonce words. This does not only hold for hidden features. It can be easily extended to inferring inflected word forms. These tasks are defined in chapter 6.

Gender assignment studies tell us that new or nonce words receive gender based on surface (or otherwise known) features, or based on their similarity to known words (Schwichtenberg and Schiller 2004). The two views are interchangeable: Surface features are an intermediate layer between a single item and the training data or exemplar pool. Similarity is the sharing of features. Therefore the models of classification (and all models going beyond that) in this thesis are based on similarity functions between linguistic items.

In the following sections, I discuss models for classification and production tasks.

1.3 Main theses

Let me divide the theses put forward here into three parts: assumptions, which I will briefly motivate, yet not test; hypotheses to be examined; and claimed contributions to machine learning for linguistics.

1.3.1 Assumptions

The main work of this thesis is only meaningful given a few assumptions. They are argued for elsewhere, and although the reader may not share all of them, I kindly ask him[5] to adopt them for the time reading.

Human language performance is based on analogy.

Read this as "certain parts of human language performance". By *language performance* I will understand any operation applied to a linguistic item, be it the assignment of a class to a known or a new word or the free production or re-production of an inflected word form. This is the performing of mappings.

Other mappings, such as translating into another language, turning a statement into a question, basically everything where two layers of linguistic annotation are involved, are possible, but the explanatory value of analogy is likely to be low here.

Analogy is to be modeled via surface similarity.

Analogy needs at least four items to be instantiated: Two surface forms which each stand in a certain relation to an item on a hidden layer. The surface and hidden forms are part of a linguistic space, and thus related within that space by a measure of similarity. In classification and production tasks the hidden features or correct outputs are to be predicted. They are therefore not accessible as features of the input. Any model of analogy must be based on the surface layer.

[5]Male pronouns are used in this thesis for simplicity and shall refer to men and women alike.

Similarity can be modeled by the sharing of features.

This is just the state of the art in data-oriented linguistic modeling. Features in this sense are either abstract properties of the data items, or concrete, primitive/atomic sub-structures.

Kernels are a mathematical way to compute measures of similarity.

As claimed by the advocates of kernels (Schölkopf and Smola 2002, ch. 1).

Kernel methods are applicable to language.

This applicability is the starting point of my thesis, and it is one side of the meeting of method and data: It is *possible* to use kernels. The other side is the appropriateness of their use in linguistics. This not a prerequisite for writing this thesis, but rather its main hypothesis to be tested.

This thesis is not an overview of kernel methods applied outside linguistics. The usual application of kernel methods is in classifications, and these also exist in linguistics. See section 3.2 for an application. I proceed from there to show that more than just classifications can be modeled by kernel methods.

1.3.2 Hypotheses

I will argue for the following main hypotheses.

Kernel methods are currently the appropriate way to abstract over exemplar models.

Linguists build similarity-based models (such as exemplar models), and similarity is based on the sharing of features (as assumed above). Other models bypass the notion of similarity and use explicit feature listings directly (see the reviews in section 2.1).

As stated above, the applicability of kernels is a necessary, yet not sufficient condition, to actually apply them. To make it sufficient, I need to point out that kernels are the state-of-the-art for modeling similarity in machine learning.

Using a kernel function has advantages over using features explicitly. First of all, such models need only the kernel as a black box, no representation of the features. Kernels are very versatile. Even large lists of possible features are easy to be handled. In this sense, the algorithm for a given kernel resembles an algorithm for multiplication of sparse vectors.

Kernels enjoy some closure properties (appendix A; Shawe-Taylor and Cristianini 2004, p. 75). This allows combining evidence from multiple layers of annotation, each equipped with an appropriate kernel. Although I do not explore these ways to enrich kernels in this thesis, it shows the flexibility the researcher has in defining a kernel.

Kernel Principal Component Analysis builds useful models.

Kernel-based models are close to connectionism and exemplar models, as shown by Ashby and Alfonso-Reese (1995) and Jäkel et al. (2008). Therefore it is reasonable to try to abstract from them using a kernel.

A kernel itself is not a model of a data set. But the feature space it induces is. In order to access the feature space and define objects (decision margins, or a basis), a kernelized factor analysis is needed. One of them is Kernel Principal Component Analysis (Schölkopf and Smola 2002). It seemed a logical choice, although its main purpose, identifying the principal components of the data set, turned out to be

In this thesis, I build reasonably competitive models with KPCA. Sometimes, performance could be boosted by using a larger data set, but the computational effort prevents this.

PCA extracts interpretable features.

PCA is meant to *extract features* (or *factors*) in form of the principal components. These are not the hidden or abstract features pre-specified by the linguist. A priori one could assume that the principal components do define similar features, and this assumption will be tested. It will turn out that interpreting the extracted factors is hard; I claim that they are meaningless for theoretical linguistics (see section 2.3.4.3). Other possibilities are briefly discussed in the outlook (section 8.2).

Chapter 4 discusses various methods to rotate the factors in order to facilitate interpretation. This achieves lower-dimensional models for classification tasks (as exemplified in 4.4), but still leaves the factors linguistically meaningless. Thus this hypothesis will be refuted.

1.3.3 Contributions

I intend to contribute the following methods and models to the field of data-oriented linguistics:

How to use kernels in linguistics.

While kernels have been previously used in linguistics (for a comprehensive overview see section 2.3), this thesis offers a wide range of possible models with kernels, with several first-of-their-kind applications. Besides the usual classification tasks, it is possible to use kernel-based models for inflection and other linguistic processes.

Rotations for Kernel PCA.

Factor rotations are standard practice, at least for non-kernelized methods. VARIMAX (Kaiser 1958), the most popular rotation method, maximizes the variance of the explicit factor loadings. Yet with kernels, these are no longer directly accessible. To my knowledge rotations for Kernel PCA have not been discussed yet. In chapter 4 I will do that.

Competitive, state-of-the-art models.

The models built for the various linguistic tasks (German gender assignment, 3.2; German nominal plural, 6.4; English past tense, 6.5) are purely data-oriented, mathematically precise and perform competitively compared to existing models in the literature.

Chapter 2

Methods

This chapter presents the most important methods for building models of analogical proportion. Section 2.1 deals with models operating on explicit feature representations. These can be circumvented by the use of a kernel function (2.2). Machine learning algorithms that can be re-formulated in terms of a kernel are discussed in section 2.3. The final section (2.4) is devoted to the main method of this thesis, *Kernel Principal Component Analysis* (Schölkopf and Smola 2002), and an extension to it, *Weighted KPCA* (Wang et al. 2005).

2.1 Explicit feature methods

The formalization of linguistic phenomena began with rule-based models, in the spirit of Chomskyan linguistics. Models were optimized with respect to storage: minimal sets of non-redundant rules. It is no coincidence that this reflects the computational capabilities of the time. In the following decades, psychologists working on cognitive models (mostly of classification) abandoned rules, suggested prototypes, and finally dropped parsimony altogether and accepted that human cognition is based on a vast episodic memory (*exemplar theory*; see Nosofsky 1986). This knowledge is modeled as a network not unlike (a simplification of) the brain, in which exemplars and their properties are linked. This line of research is known as *connectionism* (Rumelhart and McClelland 1986, see section 2.1.3). Again, this goes along with the development of computers. With the advent of large digital data sets and matching computational power, the findings from psychology could finally be incorporated into linguistic models. Since around 1990, this is the predominant approach for similarity-based models in linguistics. It is still true today that models are designed to utilize current computer technology. The present thesis is no exception to that.

I will come back to linguistic proportions other than classifications such as inflection, where the output are fully-specified word forms. Unlike inflectional *classes*, these are not commonly modeled by analogy and therefore especially interesting. Classification[1] is a much more widely known and well-studied task. It serves well for the purposes of an introduction.

The study of classification usually is mediated via language, simply because the experimenter

[1] Or *categorization*, as it is also called. The terms are usually interchangeable.

and the subject communicate. There are ways to circumvent language here, if deemed necessary. The direct way is to name classes of objects. The mental model is then probed by asking the subject to either name objects in the real world ("What is this?") or to verify classifications ("Is this a . . . ?"). The denotation of the name is the real-world extension of the class. As to the intension, the mental model that determines classification, models diverge.

Some models assume rules. They derive the class of an item from its observable properties, with the exception of a default rule, which has an empty antecedent. Such an explanation typically identifies reliable patterns. E.g., one model of German gender (Köpcke 1988; discussed in detail in section 3.3.1) makes use of selected patterns, and phonological, morphological and semantic features. As I will argue, these patterns already blur the distinction between rules and data-oriented models.

Some models derive rules from data. In machine learning, decision boundaries are learned (see section 3.3.3). Other models assume more concrete representations. An item is compared to *prototypes* of the classes, or, in the case of a full *exemplar* model, to all stored instances of all classes. A prototype is an abstraction over the set of all class members. An item is assigned to the class to whose members it is most similar. Exemplar models are discussed shortly, in section 2.1.1.

More recent models, e.g. *neural networks*, bypass the level of items and learn the mapping from observable properties to class labels. Typically, there is an input node denoting the presence or absence of each property, and an output node for each class. Network models are discussed in section 2.1.3.

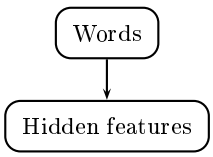
In exemplar and network models, the outcome is not discrete: All classes are activated to some degree. This is in line with human subject agreement rates, or assignment probabilities. All these models are instantiations of analogical learning: learning by example, according to pairwise similarities of exemplars. Perceived similarity is grounded in shared surface properties, and these properties also form the antecedent of classification rules. In that sense classification is *always* a mapping from observable properties to classes.

I now proceed to exemplar models.

2.1.1 Exemplar models

An analogy-based model has two layers: input and output. As a first, conceptually simple instance I choose words as inputs and grammatical features as output.

(2.1) The task: from words to their hidden features

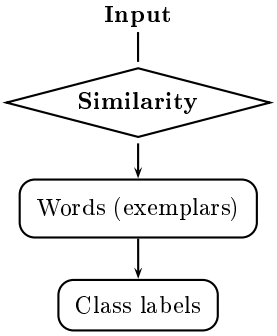


In a psycholinguistic sense, a given word form is *associated* with certain abstract features, marking the difference to its base form. It is not necessary to assume that the native speaker is aware of

a combination of a stem and a suffix, or that he derives the morphological features via any sort of rule. That is why Baayen et al. (2011) speak of *amorphous morphology* (see section 2.1.5). Still, the listener will be able to avoid clashes of grammatical features (and thus obey congruence etc.), so it is necessary to assume that he has some (subconscious) conceptualization of them, and the encounter of a word form will, subconsciously, activate these concepts. He is also able to correctly treat previously unseen word forms, and even novel word formations. This treatment is not random, but a generalization of his implicit knowledge of the language.

Linguistic items are more likely to be treated analogically if they are *similar*. This requires a notion of similarity. Classifying a new data item further requires a set of data items (*exemplars*) and an algorithm to turn the set of pairwise similarities to a class output. Nosofsky (1986) can serve as a reference for exemplar models. The architecture is depicted in 2.2

(2.2) Exemplar models



Let R be the set of all data items (of a certain linguistic type), and $(a_1, \ldots, a_n) \in R^n$ a vector of exemplars in R. Let $\{c_i\}$ be a classification of R based on a similarity function $f\colon R \times R \to \mathbb{R}^{\geq 0}$, i.e. f yields a similarity value for any pair of items, and for any item $x \in R$ there is a unique class c_i that contains x. The probability of an item $x \in R$ being assigned class c_i is given by:

(2.3)
$$p_{c_i}(x) = \frac{\sum_{j\colon a_j \in c_i} f(x, a_j)}{\sum_{j=1}^{n} f(x, a_j)}$$

p_{c_i} is the similarity to all instances of class c_i normalized by the overall similarity. As a response for x, the class with highest probability is chosen:

(2.4)
$$c(x) := \arg\max_{c_i} p_{c_i}(x)$$

In Nosofsky's formula there also are bias weights for each class. I leave them out, because they have no counterpart in the models to be discussed later on. Within the exemplar model however they are necessary to discount frequent classes, as the model is highly sensitive to frequencies in the input. Every item $x \in R$ will be mapped to exactly one class $c(x)$, therefore the space R is divided into regions corresponding to each class. More frequent classes will be stronger attractors, if not

discounted. Similarly, weights could be attached to each item in $\overrightarrow{a}$. The class weights are then just a special case, with equal weights for items of the same class.

In Nosofsky (1986) — and throughout the psychological study of classification (see Shepard 1987) — there is a division of labor between the classification algorithm (see the definition in 2.3) and the measure of similarity. Alternative classification algorithms are *prototype* models, where each class is represented by its mean, and *decision boundaries*. Nosofsky and Zaki (2002), Nosofsky and Stanton (2005) discuss their relation to exemplar models and conclude that the latter are more adequate.

What remains to be specified is the similarity function f. Earlier experiments had suggested that similarity decays asymptotically with perceived distance. Nosofsky (1986) thus derives similarity from a distance metric d, via an exponential or a Gaussian function:

(2.5)
$$\forall x, y \in R\colon f(x, y) := e^{-d(x,y)^s}$$

with $s = 1$ for the exponential and $s = 2$ for the Gaussian function.

In order to compute distance, Nosofsky employs a 'multidimensional perceptual representation', that is essentially a vector of explicit perceivable features, which are possibly weighted. Each feature corresponds to one dimension in the multidimensional representation. The distance on a single dimension simply is the difference of the values two items take. This crucially assumes numerical values.[2] The aggregate value over m features is calculated with a Minkowski r-metric:

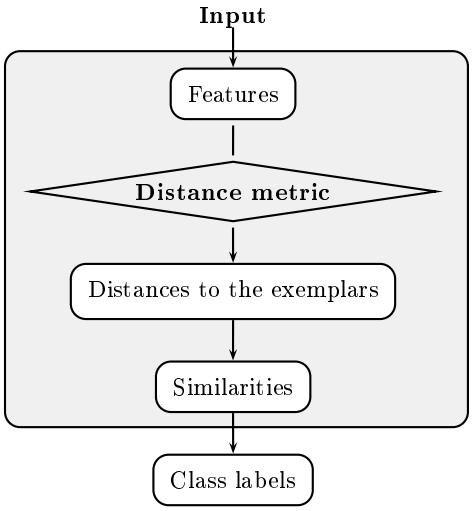
(2.6) Let $\phi(x)$ list the features of any $x \in R$, such that $\phi_k(x)$ is its kth feature.

$$\forall x, y \in R\colon d(x, y) := \left(\sum_{k=1}^{m} |\phi_k(x) - \phi_k(y)|^r \right)^{\frac{1}{r}}$$

Again, I left out the feature weights. The Minkowski r-metric is a generalization of distance measures. It represents Euclidean distance with $r = 2$ and the city block metric with $r = 1$. Thus distance is an aggregate value over explicit feature representations, and similarity is a function of distance.

(2.7) Exemplar model with distance/similarity function

[2]The treatment of other data types is an instance of a kernel, see section 2.2.3.

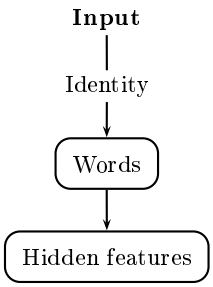


Exemplar models have been and are successful in describing human classification behavior. In the following I discuss several possibilities to abstract over exemplar models. I call a method an abstraction if the model is computed, not trained, and if the final model does not refer to intermediate layers. One of them is found in artificial neural networks (sec. 2.1.4). Another leads to Naive discriminative learning (Baayen et al. 2011), discussed in section 2.1.5. And finally, Kernel Principal Component Analysis (Schölkopf and Smola 2002; sec. 2.4) is used in this thesis as such an abstraction of exemplar models.

2.1.2 Spreading Activation

The process of inferring features can be described in terms of activation spreading through a network: Surface features are activated by listening, and they activate nodes representing higher-level features. The simplest spreading activation model is as follows: Associate each word in the training data with all its features. This model will correctly recognize known words, but it cannot handle unseen words and will grow proportionally to the number of seen words. It fails to generalize.

(2.8) Lexical lookup of hidden features

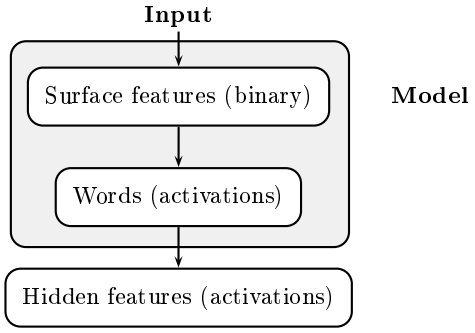


A first step in generalizing is to treat an unseen word analogically, that is, to treat it as similar words are treated. Given a measure of similarity between words, each known word will receive activation in proportion to similarity, and pass it on to their features. The result will not be exact, since not all words will activate the same features. The result will rather be a distribution of activation over all features, with the relative activation strengths depending on the similarities. As an example,

a word being more similar to plural words will get more activation for *plural* than for *singular*. Thus an unseen word can be classified by analogy, based on similarity.

The next step is to explicate similarity. Storing pairwise similarities of words cannot be used for generalization, since the similarity values would need to be known. In order for *similarity* to work associatively (by spreading activation), it needs to be based on *identity* at a lower level, a level of surface features. For now take as an example unigrams and bigrams: substrings of length 1 or 2. In section 2.2.3.1, I discuss different possibilities of surface features for words.

(2.9) Activation spreading through shared surface features



The complete spreading activation model now consists of the following layers: input, surface features, seen words, hidden features. The input word activates all its unigrams and bigrams as features. It is straightforward to decompose a word in such a manner. Unigrams and bigrams then activate the pool of known words, resulting in an activation distribution over the pool: each word will receive as many incoming active connections as it shares surface features with the input word. Activation is passed on to the layer of hidden features, so the more similar words have greater influence. One could normalize over mutually exclusive features such as singular/plural, resulting in an activation distribution over the potential values. Features are thus recognized in a fuzzy way.

Here one needs to make a distinction: for mutually exclusive features, choose the value with the highest activation. For others, set a threshold (cf. appendix D.1). Further processing could operate on either the fuzzy representation (in order to resolve the ambiguity at a higher level) or on the disambiguated one.

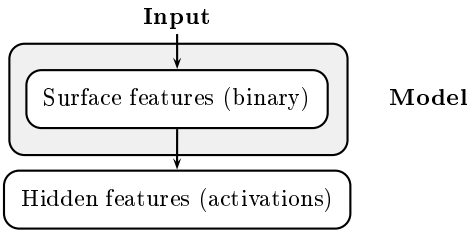
This process of analogy is cumulative: Every training item will add its contribution. This bears some disadvantages. First, frequent features will be overrepresented. Second, even items from the training set are not mapped uniquely to their features. Consider testing a training item. Similarity to itself is maximal, but that does not give us the same morphological features the item originally had, because it also enjoys similarity to other words, which in turn add up to the features. The inferred features from an already seen item depend on the rest of the pool. Thus recognition is imperfect.

A possible solution is to train weights for every connection in the network. Training weights means to learn the influence each surface feature has on each hidden feature. Thereby the layer of words can be circumvented (sec. 2.1.3): activation can spread directly from the surface layer to the layer of hidden features. Adding intermediate layers then leads to artificial neural networks (sec. 2.1.4).

2.1.3 Connectionism

The network architecture without a layer for words is depicted in ex. 2.10. It bypasses the layer of words, and directly learns weighted connections from surface to hidden features. The input is encoded, that is, mapped to a feature vector with binary values, where each position in the vector can activate one input node of the network. Activation spreads according to the weights, such that each output node receives an activation value (in $\mathbb{R}$).

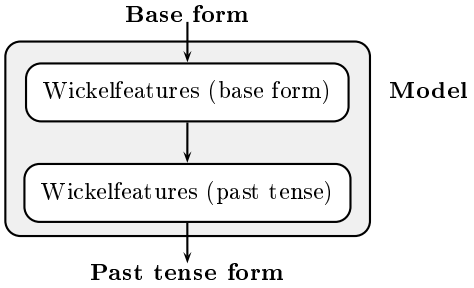
(2.10) Connectionist network



The weights are trained from a training data set in order to faithfully associate all training items with their correct features. Networks can also assign outputs to unseen inputs (not from the training set), and they allow for these inputs to be treated in analogy to the the training items. This is in line with the native speakers' abilities.

A classical instance of such a model is devised by Rumelhart and McClelland (1986) for learning the past tense of English verbs (also see section 6.5). It first encodes base forms in a 'Wickelfeature' representation (after Wickelgren 1969), which I will explain shortly. This layer is fully connected to a layer consisting of the same features, representing the past tense form, which has to be further decoded into an actual word. The architecture is shown in ex. 2.11.

(2.11) Rumelhart and McClelland (1986)



The network is formed by the two inner layers; the encoding and decoding steps are separate. The Wickelfeature representation is rather complex. Initially, the input is written as a sequence of 'Wickelphones'. These are phonemes in context: one phoneme in each direction, where a word edge is denoted by a sentinel #.

(2.12) Wickelphone representation for /kæt/ (*cat*):

$\{\ {}_{\#}\mathrm{k}_{\mathrm{æ}},\ {}_{\mathrm{k}}\mathrm{æ}_{\mathrm{t}},\ {}_{\mathrm{æ}}\mathrm{t}_{\#}\ \}$

Essentially, these features are *trigrams*. In theory, the network is supposed to learn patterns such as ɪŋ# → æŋ# for a certain subclass of irregular verbs, and mappings to əd# etc. for regular verbs. The practical problem is that there are over 40,000 possible trigrams, resulting in more than 1.6×10^9 weights to be learned. This is highly impractical. Most important, it has far too many degrees of freedom.

Rumelhart and McClelland therefore devise a set of 10 binary phonetic features to describe every phoneme,[3] plus a marker for the sentinel #. They then map trigrams to feature triples (e.g. `stop - vowel - stop` for *cat*), and cut down the set of feature triples to 460. Finally, a word is represented both on the input and the output side by the presence or absence of each of these 460 feature triples. Considerably fewer weights ($460^2 = 211600$) have to be learned. The model is trained on 420 high- and medium-frequency verbs of English, and tested on 86 low-frequency verbs. It achieves to correctly predict 90% of the feature triples for regular verbs, and 85% of the irregulars.

Now 90% of the features do not yet make an output. This representation has to be decoded. In Rumelhart and McClelland's algorithm for doing so, trigrams compete to explain feature triples. The more feature triples a trigram covers, the safer it is to assume that it actually appears in the output. These features become unavailable for the competing trigrams, which continue to compete for the remaining features. This process can eliminate incompatible features and lead to a cleaner representation in trigrams, which can be evaluated against the desired output in terms of correctly predicted trigrams.

These trigrams are not in any particular order, because sequential information is only inherently stored in the trigram representation. With a further restriction that the trigrams have to form a consistent sequence, fully specified forms can be returned for previously unseen verbs. This is an instance of *pre-imaging*, a concept which will be discussed later (sec. 2.4.4): finding a discrete output when all that is available is an activation over features.

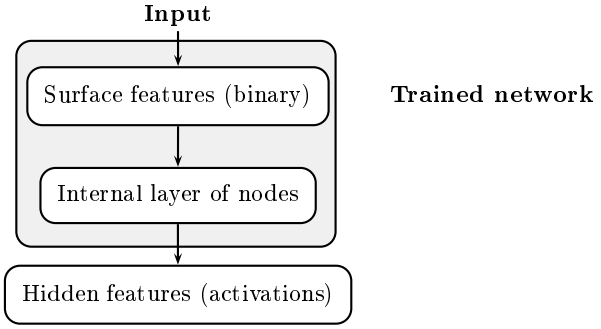
The Wickelfeature representation's complexity is due to the need to reduce the number of weights when training a network. An alternative input representation, which is more direct and yet needs fewer input nodes, is shown in the next section; a generalization for which the number of features is irrelevant is found with kernels (section 2.2).

2.1.4 Artificial neural networks

Ways to reduce the number of weights to be trained include imposing restrictions on the connections, and cumulating information. Cumulation is achieved by fully connecting part of the input nodes to a set of internal nodes. These are in turn connected to the output. There may be any sort of internal architecture, although for computational reasons one usually restricts models to *feedforward networks*. These do not have cycles, and there exist efficient algorithms to train the weights. The architecture is shown in ex. 2.13, the difference to the previous model being the internal layer of nodes.

[3] Ten features do not totally suffice, leading to a few identically represented phonemes.

(2.13) Artificial neural network



Artificial neural networks are a way to abstract over exemplars, and therefore an instance of connectionism. With internal layers, they even abstract away from the original input features. An internal layer is supposed to contain a compressed representation of the input, from which the extraction of hidden features can be learned much more efficiently. The output layer usually comprises one node per abstract class to be activated. Such a network is exemplified by MacWhinney et al. (1989), discussed in 3.3.2.

The input in networks such as the one used by MacWhinney et al. (1989) is represented by hand-selected features, or by a general scheme such as the Wickelfeatures. Another approach is the one taken by Plunkett and Nakisa (1997) and Hahn and Nakisa (2000). They employ a template of consonant and vowel slots. Let us call this a *CV-template*. See example 2.14.

(2.14) Word-to-template alignment example for the German noun "Quaste" (*tassel*) taken from (Hahn and Nakisa 2000):

```
VCVCVCVCVCVCVCVC
_________k_v&s_t@_
```

Each symbol in the phonemic representation stands for a vector of 15 binary features, yielding 240 binary input nodes in total. Empty slots (_) are represented by a vector of zeros. Note that the word is right-aligned within the template, leaving some slots open. This representation is much more legible than the Wickelfeatures, it contains total information on the order of phonemes, and it needs only half the input nodes.

Hahn and Nakisa (2000) map German singular nouns to binary vectors via the template, and train a network to classify them into 15 German plural classes. I will come back to their model in section 6.4.7.

Right-alignment has one problem besides the empty slots: When suffixing a vowel, all preceding phonemes have to shift to the left. Plunkett and Nakisa (1997) use left-alignment. The problem of aligning phonemes to slots is discussed by Bullinaria (1997). In some template approaches (see below), there are not enough slots, so some phonemes are dropped from the representation. Such templates are a mix between a faithful representation of the input and hand-selected features.

Plunkett and Nakisa (1997) employ a template similar to the one in ex. 2.14 for Arabic plural formation. They find that the phonological form of the singular is a good predictor of plural type.

For plural formation, they train a neural network. 95.8% of the (859) seen words could be predicted correctly by a network with 200 hidden units.

Plunkett and Juola (1999) use a CCCVVCCC template for monosyllabic verbs and nouns, with center-alignment, to predict plurals for nouns and past tense forms for verbs. With 16 binary phonetic features to describe every phoneme of English, there are 128 input nodes, and the same amount of output features. If there are two syllables, the first one is dropped. See the further discussion in sec. 6.5.2.

The actual output of such a network is a vector of relative activations of the output nodes. This is interpreted slot by slot: For each slot in the template, the nearest proper phoneme is found, including the empty phoneme (_). This again is a form of pre-imaging (see above).

The models of Rumelhart and McClelland (1986), Plunkett and Nakisa (1997), Plunkett and Juola (1999) allow for returning fully specified output forms, not only activations of classes. They thus instantiate paradigmatic analogical proportion (Albright 2008, Pirrelli and Yvon 1999). This is achieved by mapping words to binary vectors, learning a mapping to a likewise representation of the output, and mapping this back to a word.

In this thesis I suggest that both on the input and on the output side kernels simplify matters greatly by abstracting over the explicit feature vectors and their alignment problems. This also allows for variable word lengths, other data structures than words, and more flexible and elegant pre-imaging (see sec. 6.2.1). Instead of networks, a kernelized factor analysis is needed (sec. 2.4).

The disadvantages of networks are in the input representation (the problem of aligning phonemes to slots, see Bullinaria 1997), their rather arbitrary internal architecture (one has to choose the number of layers and hidden nodes) and in the iterative training process: At the first shot, a network does not even model the training data. There are mathematically more appealing alternatives.

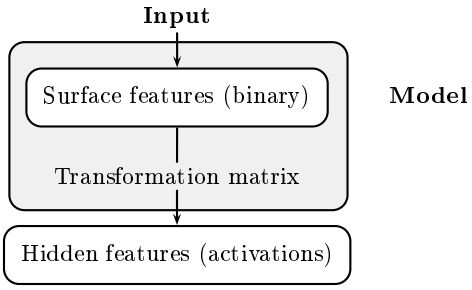
In terms of classifications, there also is one model to circumvent the need to train weights, to be discussed next: a method which directly computes the weights in a setting similar to the one discussed above (sec. 2.1.3).

2.1.5 Naive discriminative learning

Naive discriminative learning (Baayen et al. 2011) abstracts over the intermediate layer of seen words. The task is to map inputs onto outputs. Inputs are words, or, for practical purposes, their representation on the second layer: an activation vector over unigrams and bigrams. Each position in the vector denotes presence (1) or absence (0) of a feature. Alternatively, it could denote the number of times the feature occurs. The first option is used by Baayen et al. (2011), whereas I will prefer counts.

Only two layers remain: surface and hidden features, called *cues* and *outcomes*. The mapping between the two can be captured in a single matrix of cue-to-outcome weights. This bypasses the word layer.

(2.15) Mapping from surface to hidden features (Baayen et al. 2011)



2.1.5.1 Method

Naive discriminative learning builds upon the asymptotic final state of a model of learning. It thus bypasses the whole learning process, and directly computes the final weighting matrix. The learning model is the *Rescorla-Wagner model* (Wagner and Rescorla 1972), a popular and successful model in psychology. In each step, the model is presented with a data item comprised of cues and outcomes (surface and hidden features). The association strength (the weight) between a cue and any outcome increases whenever the outcome is also present. Regardless of the presence of the outcome, association strengths decay for each cue present. This decay depends on other cues competing to explain the outcome: The more competitors, and the stronger their association, the higher the decay.

This model changes with every data item, and thus does not converge in general. However a stable state can be defined such that the *expected* change (by randomly drawing a data item from the data set) for all weights is 0. This *equilibrium* state of the Rescorla-Wagner model was derived by Danks (2003), and its characterization is surprisingly simple. I adapt the definitions from Baayen et al.'s appendix in appendix F. A short overview is presented here.

Assume a set of cues (surface features) and a set of outcomes (hidden features). Let C be a matrix of cue co-occurrences: $C_{i,j}$ is the number of times the cues c_i and c_j appear together in a data item. Note that these are data tokens, not just types. Frequencies are easily incorporated (see appendix F). Similarly, let O be a matrix of cue-outcome co-occurrences: $O_{i,j}$ is the number of times cue c_i and outcome o_j appear together. The weighting matrix W now is the solution of $CW = O$.[4] Cues and outcomes are fully connected, with W defining the weights between every cue and every feature. This corresponds to a simple network with one input layer (surface) and one output layer (hidden features; see figure 2.15).

For a testing item, the vector of outcomes is now computed by multiplying the cues (the surface feature vector) by W. The matrix correctly maps the training data if the chosen surface features are expressive enough.[5] If not, and for unseen testing data, the activations of hidden features are to be seen as relative to each other; the highest ones are to be assumed present (cf. the pre-imaging already mentioned in section 2.1.3)

The method is called *naive* because the outcomes are implicitly assumed to be independent. They are not conditioned on each other.

[4] This can be computed via the inverse of C if C is invertible, or as its pseudoinverse.

[5] That is, again, if C is invertible.

2.1.5.2 Walkthrough

The method is easy to implement, given a way to enumerate the features of a word. The remainder of this section walks through the original toy example in the appendix of Baayen et al., in a little more detail.[6] There are explicit surface (2.16) and hidden (2.17) feature vectors. Additionally, each word comes with a frequency (2.18). F is a square matrix with the frequencies on its diagonal, and 0 otherwise.

(2.16) Surface features (unigrams) A:

	hand	hands	land	lands	and	sad	as	lad	lads	lass
a	1	1	1	1	1	1	1	1	1	1
d	1	1	1	1	1	1	0	1	1	0
h	1	1	0	0	0	0	0	0	0	0
l	0	0	1	1	0	0	0	1	1	1
n	1	1	1	1	1	0	0	0	0	0
s	0	1	0	1	0	1	1	0	1	1

(2.17) Hidden features B:

	hand	hands	land	lands	and	sad	as	lad	lads	lass
and	0	0	0	0	1	0	0	0	0	0
lass	0	0	0	0	0	0	0	0	0	1
sad	0	0	0	0	0	1	0	0	0	0
as	0	0	0	0	0	0	1	0	0	0
land	0	0	1	1	0	0	0	0	0	0
plural	0	1	0	1	0	0	0	0	1	0
lad	0	0	0	0	0	0	0	1	1	0
hand	1	1	0	0	0	0	0	0	0	0

(2.18) Word frequencies f; F is a diagonal matrix containing these values.

hand	hands	land	lands	and	sad	as	lad	lads	lass
10	20	8	3	35	18	35	102	54	134

Given all these, the model calculates the weights W from $AFA^T \times W = AFB^T$ (2.19).

[6] There is a minor difference. In the latest available version of their paper, a count of 0 s for 'lads' underlies the subsequent calculation. This may differ from the final, published version of Baayen et al. (2011). Therefore my matrix W, given in table 2.19, will differ from theirs.

(2.19) Weights W:

	and	lass	sad	as	land	plural	lad	hand
a	0.208	0.274	0.066	0.726	-0.208	-1.066	-0.066	0.000
d	-0.024	-0.798	0.226	-0.202	0.024	0.774	0.774	0.000
h	-0.759	0.078	-0.163	-0.078	-0.241	0.163	0.163	1.000
l	-0.171	0.545	-0.284	-0.545	0.171	0.284	0.284	0.000
n	0.623	0.389	-0.235	-0.389	0.377	0.235	-0.765	0.000
s	-0.072	0.087	0.159	-0.087	0.072	0.841	-0.159	0.000

Features are reconstructed from surface forms via $B' = A^T W$ (2.20).

(2.20) Reconstructed hidden features B':

	hand	hands	land	lands	and	sad	as	lad	lads	lass
and	0.048	-0.024	0.635	0.564	0.807	0.112	0.136	0.012	-0.060	-0.036
lass	-0.058	0.029	0.409	0.496	-0.136	-0.437	0.361	0.020	0.108	0.906
sad	-0.106	0.053	-0.226	-0.067	0.058	0.451	0.225	0.008	0.167	-0.059
as	0.058	-0.029	-0.409	-0.496	0.136	0.437	0.639	-0.020	-0.108	0.094
land	-0.048	0.024	0.365	0.437	0.194	-0.112	-0.136	-0.012	0.060	0.036
plural	0.106	0.947	0.226	1.067	-0.058	0.549	-0.225	-0.008	0.833	0.059
lad	0.106	-0.053	0.226	0.067	-0.058	0.549	-0.225	0.992	0.833	0.059
hand	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

First note that 6 surface features are not enough to fully capture the data. Even for the known data points, recall is not perfect. The surface feature `h` uniquely identifies the lexeme *hand*. All other values are mere approximations.

The lexical hidden features could extend to other inflectional forms. The abstract feature *plural* is heavily influenced by the small data set: `s`, the plural marker, is not even a reliable cue to plural.

Remember that surface features are either present or absent, and thus take as values 0 or 1. Changing this to occurrence counts only gives `lass` a count of 2 `s`. This facilitates the recognition of words containing `s`, but hinders the recognition of plural (see 2.21), because the feature `s` is now even less of a cue to plural.

(2.21) Reconstructed hidden features, giving `lass` a count of 2 `s`:

	hand	hands	land	lands	and	sad	as	lad	lads	lass
and	-0.008	0.004	0.635	0.647	0.799	0.154	0.121	-0.022	-0.010	-0.032
lass	-0.161	0.080	0.320	0.561	-0.121	-0.254	0.252	-0.054	0.187	0.934
sad	-0.153	0.077	-0.315	-0.086	0.079	0.592	0.131	-0.032	0.197	-0.034
as	0.161	-0.080	-0.320	-0.561	0.121	0.254	0.748	0.054	-0.187	0.066
land	0.008	-0.004	0.365	0.353	0.201	-0.154	-0.121	0.022	0.010	0.032
plural	0.314	0.843	-0.005	0.524	0.042	0.663	-0.383	0.087	0.616	0.100
lad	0.153	-0.077	0.315	0.086	-0.079	0.408	-0.131	1.032	0.803	0.034
hand	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

2.1.5.3 Applications

Baayen et al. (2011) model self-paced reading times and visual lexical decision in Serbian and English. Cues are unigrams and bigrams, and outcomes are both lemmas and morphological features (case and number, for Serbian). The models maximize the expected probabilities of outcomes given the cues. Delays in reading and identification are successfully attributed to competing outcomes.

They call the model an instance of *amorphous morphology*, because morphological features are inferred from surface forms without the intermediate notion of a morpheme. This allows for modeling allophonic realizations of a morphological feature (one feature to many different exponents) as well as homophones (many features to one surface form).

Baayen (submitted) develops a model for the English dative alternation. In English, dative constructions are either of the type "gave Mary a book" (two NPs) or of the type "gave a book to Mary" (NP-PP). The choice is known to depend on definiteness, animacy, the pronominal status of the constituents, their lengths and others. Baayen uses these as cues, and the two constructions as outcomes. The model's predictions are 92% accurate.

2.1.5.4 Conclusion

Naive discriminative learning is superior to networks in that it does not need training of weights. It also incorporates frequencies, but is inflexible with regard to the choice of surface features because they need to be listed explicitly. The cue-by-cue matrix could become prohibitively large.

In 2.4.6, I compare the model discussed here to a kernel-based model of the same toy data.

2.2 Kernels

2.2.1 Similarity functions

In exemplar models, a similarity function f is used to calculate similarity values for any pair of items (see section 2.1.1). Nosofsky (1986) defined similarity in terms of surface features. Likewise, network models (section 2.1.4) represent inputs as feature vectors. The consensus is that the *similarity* of complex, structured objects can be defined in terms of *identity* of their parts or otherwise observable features.

The Wickelfeatures of Rumelhart and McClelland (1986), the CV-templates of Plunkett and Nakisa (1997), Plunkett and Juola (1999), Hahn and Nakisa (2000), and the matrices of Baayen et al. (2011) are designed to satisfy a constraint for concise representations. In machine learning, one usually does not know beforehand which features will be informative, so it is a general strategy to use as many as are extractable from the data. Instead of representing a data item within the model as a vector of m surface features, it can be treated as a vector of similarities to n training items. Given $m > n$, the latter option is more feasible. Similarity functions hide the explicit feature vectors, and thus are preferable whenever the number of features is large.

2.2.2 Definition of kernels

Kernels are a general class of similarity functions (Schölkopf and Smola 2002, Shawe-Taylor and Cristianini 2004), which typically build upon surface features. Let $\phi\colon R \to \mathbb{R}^m$ be the *implicit mapping* of data items to the m-dimensional vector representing their surface features. A kernel $k\colon R \times R \to \mathbb{R}$ is defined as the inner product of feature vectors: $\forall x, y \in R\colon k(x, y) := \langle \phi(x), \phi(y) \rangle$.

Consider the example from the previous section (2.1.5.2). The matrix A in 2.16 gives the surface features for all data items: The ith column in A is the feature vector $\phi(a_i)$. The kernel defined by ϕ is calculated as $k(a_i, a_j) := A^T_{1:m,i} A_{1:m,j}$. The matrix of all kernel values $K = A^T A$ is called the *Gram matrix* (or *kernel matrix*). For the example, it is given in 2.22.

(2.22) Gram matrix K for the data set:

	hand	hands	land	lands	and	sad	as	lad	lads	lass
hand	4	4	3	3	3	2	1	2	2	1
hands	4	5	3	4	3	3	2	2	3	2
land	3	3	4	4	3	2	1	3	3	2
lands	3	4	4	5	3	3	2	3	4	3
and	4	3	4	3	3	2	1	2	2	1
sad	2	3	2	3	2	3	2	2	3	2
as	1	2	1	2	1	2	2	1	2	2
lad	2	2	3	3	2	2	1	3	3	2
lads	2	3	3	4	2	3	2	3	4	3
lass	1	2	2	3	1	2	2	2	3	3

The Gram matrix has the properties of being *symmetric* and *positive semi-definite*, and all symmetric positive semi-definite matrices K can be decomposed into $A^T A$, where A is a features-by-items ($m \times n$) matrix. See Shawe-Taylor and Cristianini (2004, p. 57f.) and appendix B for the full definitions and the proof. This decomposition is not unique: different feature vectors can result in the same vector products.

For every similarity function and every data set there is a Gram matrix. If this matrix is positive semi-definite the similarity function is a kernel: there is guaranteed to be a mapping ϕ such that A is the column-wise concatenation of feature vectors $\phi(a_i)$. The definition of a kernel in terms of the positive semi-definiteness of its Gram matrices is therefore equivalent to the definition in terms of its implicit feature mapping ϕ. The features are immediate if the kernel was defined via an explicit feature mapping ϕ in the first place, and in this thesis I will be looking at such cases only.

In the general case, the feature mapping is *implicit* because the features do not need to be known. For kernels obtained e.g. by combining existing kernels (see appendix A.1), features can be made explicit: the feature vectors of the sum of two kernels are the concatenation of the respective feature vectors, and the features of the product of two kernels are the pairs of the original features.

Features — as devised by the researcher — do not need to take numerical values, but they can be internally complex. For example, a phoneme can be a feature, but it is not a number. The set of

values a feature can take defines a certain value type. One can get a numerical value by applying an inner kernel formulated over this type. An inner kernel for phonemes could be defined on (binary) phonetic features. This nesting of kernels allows for treating structured data, and it is called a *convolution kernel* (Haussler 1999). Any nesting bottoms out to the *identity* of sub-features which are decomposed no further.

2.2.2.1 Conclusion

Kernels are customizable: They encode the researcher's intuition and knowledge and experimental insights about what constitutes similarity in a given domain. They characterize the surface of (linguistic) objects, their observable properties: phonemes of a word, but also any annotation that is available (syntactic structures etc.). These properties are the implicit features of the kernel. As kernels compute measures of similarity of data items, they are an essential part in analogy-based models. This is in line with exemplar models.

Kernels are a high-level approach to computing similarity, as opposed to inner products of explicit feature vectors. They allow useful data analysis and machine learning methods to be applied to structured data (2.3). Is linguistic data representable by numerical vectors, or is it rather to be called structured data? I suppose semanticists, syntacticians, phonologists and most other linguists would prefer the high-level approach, and see their work in determining the structure of linguistic objects. Sometimes, in phonetics and psycholinguistics, one actually works with numerical data (reaction times, event-related potentials, formant frequencies, ...). Even then, similarity would not most readily be captured by inner products, and even if it was, kernels fare no worse computationally because the explicit computation is always possible.

2.2.3 Kernels for linguistic data

Typical linguistic data structures are strings (words or sentences) and graphs, and among the latter especially trees. A cumulative notion of similarity for these is to use counts of shared sub-structures as features. These features do not need to be binary. In fact it makes sense to *count* occurrences of sub-structures. This allows for distinguishing items with more or fewer occurrences of a certain sub-structure.

Section 2.2.3.1 exemplifies the construction of a kernel for words, by taking the parts of a word as its features, choosing an appropriate decomposition of words into parts. The resulting kernel is a typical instance of a *string kernel*. These are the best-known type of kernel for structured data (cf. Shawe-Taylor and Cristianini 2004, ch. 11). Section 2.2.3.2 presents a kernel for trees, following Collins and Duffy (2001a).

2.2.3.1 String kernels

In many applications in this thesis and in general, the linguistic objects in question will be strings of some sort, e.g. words or sentences. Strings and sequences are also used as a means of representation

outside of linguistics, and therefore there is a large variety of string kernels available.

In the example applications to come, I need a kernel on words, capturing what makes words similar in a linguistic sense. Also refer to the definitions in appendix A.2.

A word is recognized by its surface features. Representing a word as a string, those features are its substrings/subsequences. Substrings are sequences of letters which appear in the same order as in the whole word. Subsequences are contiguous substrings, better known as n-grams in computational linguistics. One may want to restrict the features in size, say to substrings of a length equal to n, or at most n.

(2.23) As an example, take the words `tree` and `tear` and a 3-gram kernel (and read that as "up to 3"-gram: unigrams, bigrams, and trigrams). The features are:
`t,r,e,e,tr,re,ee,tre,ree` and `t,e,a,r,te,ea,ar,tea,ear`, respectively. Note that the features are not binary ("contains `e`"), but that they count occurrences.

The kernel value of `tree` and `tear` is the number of pairs of identical features. The `t` of `tree` and the `t` of `tear` are such a pair, and so are the `r`'s. The first `e` of `tree` and the `e` of `tear` are, and there is another pair with the second `e` of `tree`. The two words share no bi- or trigram. In sum, that is a kernel value of 4.

n-grams can be enumerated (primarily by length, secondarily by alphabet), thus an explicit vector representation is possible given an upper bound on the length n. The similarity of two words is the inner product of these vectors, but it can be calculated much more efficiently. The size of the vectors grows exponentially with n, while most values are 0: a word consists of a fairly small number of different n-grams. The algorithm for calculating the kernel value is given in the appendix (A.2). It does not need the explicit feature vectors to be stored and multiplied, and thus is much more efficient.

Preliminary experiments led me to favor subsequences (n-grams) over substrings. The cases studied here require local information only.[7] There is, however, not a best length for n-grams. Choosing n too large results in data sparseness (lots of unseen n-grams), choosing it too small misses relevant patterns. Unigrams, for example, don't know anything about the order of letters in a word.[8] On the other hand, unigrams turn out to be very reliable, because they are not affected by data sparseness.

For higher n-grams the situation is reversed: 4-grams easily identify known words, but characterizing unknown words only by known 4-grams easily creates a situation where every trigram of a word is well attested, but none of the 4-grams is, so that the word would have zero similarity to any other. Therefore, I will never give up the simpler, more reliable information. In practice that means, when I say n-gram kernel, I really mean *up-to-n*-gram kernel. Thus a 4-gram string kernel will have as features all uni-, bi-, tri- and 4-grams of a word. And since this is the only string kernel I use, the term string kernel will refer to n-gram kernels in particular.

[7] In other cases, non-local information could be useful: A language with frontness vowel harmony would lack vowel sequences of mixed frontness.

[8] And even higher n-grams are ambiguous in pathological cases, see lemma 6 in appendix A.2.

In the following n will often be a parameter, and I will examine its influence on model accuracy. The upper limit I set on n is 6, quite arbitrarily.

Substring features are oblivious of their position within the string. This is both a benefit, since it allows for more abstract descriptions of strings, and a disadvantage, when the position actually does play a role. It is both easy and generally useful to additionally encode the edges of a string with a sentinel. Here I use the symbol '#' (see section 3.1). n-grams covering the sentinel are positional now, but no other features are.

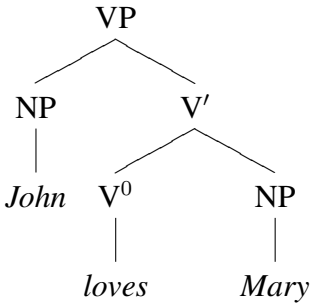
A final remark: n-grams refer to the units a string is composed of. For words, the unigrams are the letters; for phonetic transcriptions, single phonemes (with diacritics); and for sentences, words.

2.2.3.2 Tree kernels

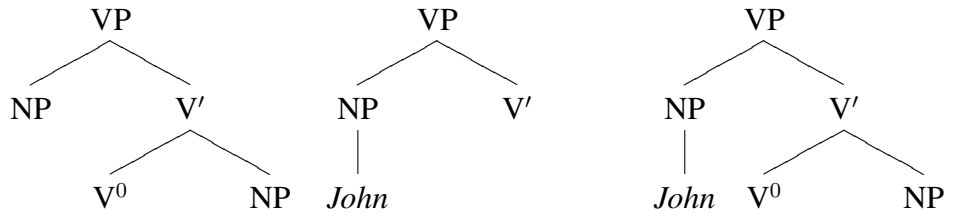
A kernel for trees can be devised in a very similar fashion (Collins and Duffy 2001b). A characterization of its substructures is needed; they will serve as the features. Data-oriented parsing (DOP; Bod 1998) employs all contiguous subtrees as substructures, with the condition that of each node, either all or none of the daughters are contained. Definitions for trees, subtrees and Collins and Duffy's algorithm for computing the tree kernel are found in appendix A.3. An example follows.

Similar to the string kernel above, the features of this tree kernel are contiguous. Corresponding to the bound n on their length, we can employ a height bound h, either as exactly h or as up-to-h. For the example, I choose subtrees up to a height of 3. Height is defined as the number of nodes in the longest path within a tree.

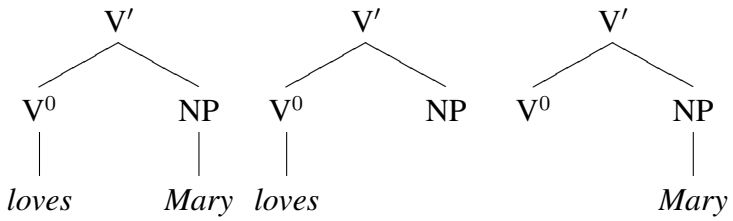
(2.24) Consider the following syntactic tree:[9]



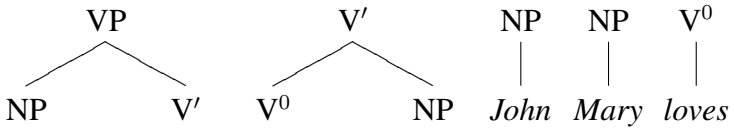
Its subtrees of height 3 are:



[9]The example is taken from the documentation of `xyling.sty` by Ralf Vogel, which was used to draw this tree. See `http://www.ling.uni-potsdam.de/~rvogel/xyling/`



Height 2 trees are context-free productions:



And height 1 tress are all nodes: *John*, *Mary*, *loves*, NP, V^0, V', VP.

The kernel then is the sum of shared features. For two subtrees to be identical, there needs to be a mapping (a homomorphism) from the nodes of one subtree to the other preserving all dominance and precedence relations. Subtrees are counted, not binary features. They could be enumerated and listed in a vector to compute the inner product, but the algorithm (see A.3) by Collins and Duffy is more efficient.

I will use this kernel in 7.2.2. A larger variety of tree kernels is discussed in Costa Florêncio (2007).

2.2.3.3 Other kernels

Examples of the linguistic objects are anything from the phoneme via words, phrases, sentences, utterances, texts, up to whole languages. In order to feed them into a kernel, formalizations of these are needed: IPA characters or phonetic feature bundles, strings of letters or of IPA characters, phrase-structure trees, attribute-value-matrices of Head-Driven Phrase Structure Grammar (HPSG), lambda-terms etc. It is the linguist's task to mathematically define similarity on objects of the same type. Following the general scheme of a kernel, this can be done by identifying sub-structures — in the fashion of data-oriented parsing (Bod 1998) — and using them as features. In a way similar to DOP being extended to other syntactic annotation schemes (see e.g. Arnold and Linardaki 2007), kernels can be defined for such structures.

2.2.4 Why linguistics needs kernels

In a nutshell, these are the reasons to use kernels in linguistic modeling (cf. Jäkel et al. 2009):

- They are an active field of research in machine learning.
- They are successful in cognitive modeling (mostly classification), and language is an important part of cognition.
- They are mathematically well-founded and generalize explicit implementations of similarity, as used in various linguistic data-driven and exemplar-based models.

- They are modular: any kernel-based method (see 2.3) can use any kernel.

To conclude: It is the right time to apply the methodology recently established in machine learning to new challenges in another field, linguistics.

In this thesis, I formulate kernel-based models for some tasks in linguistics, which have already been described using similarity as given by explicit feature representations. Naturally, this begins with the easier tasks, first of all classification, the most basic task in machine learning, which is also found on all layers of linguistic data. It is not to be expected that kernels as a new method will improve the performance of these models, because they are already optimized with respect to performance.

The current task is to accommodate the previously established knowledge in a kernel. However building kernel models for these tasks lays the ground for more complex applications, such as production tasks, which are discussed in chapter 6. Previous analogy-based models for these tasks are rare and complicated (see 2.1.4). I consider kernel methods to be the natural formulation of the same ideas with today's methodology. That is to say, their authors would have used kernels, had they been available to them. In that sense I hope for the present thesis to build a useful bridge between linguistic modeling and current ML.

2.3 Implicit feature methods

This section discusses learning algorithms operating on kernel vectors and their relevant predecessors operating on explicit feature vectors, with respect to their usage in linguistics. The main method of this thesis, Kernel Principal Component Analysis, is given a separate section (2.4).

The methods in 2.1 calculate similarity directly from explicit surface feature representations, or derive hidden features from surface feature co-occurrences. In *kernel methods* all calculations involving feature vectors $\phi(x)$ are inner products of them. The calculation is done via the kernel (see 2.2.2).

Kernel methods usually derive from an equivalent method working on explicit feature vectors. Many machine learning algorithms turned out not to need the actual feature vectors, but only the inner product in the space spanned by these features (Schölkopf and Smola 2002). Replacing the inner product by a kernel is known as the *kernel trick*.

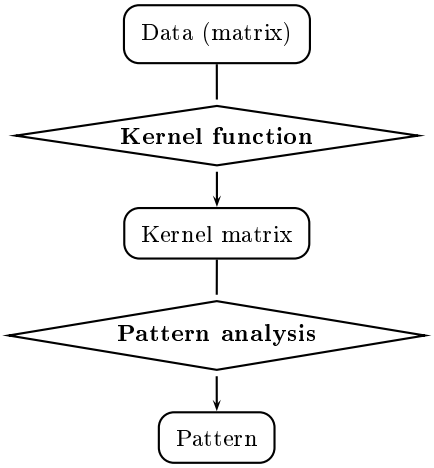
The methods I present here are applied both with and without kernels. The difference only comes into play when results need to be interpreted: then, the required (surface or hidden) features have to be made explicit.

Kernel methods are generally not specific to a certain field of application, or to a certain data type. This is crucial in bringing them to new applications, such as in linguistics. Inside the kernel are the researcher's insights and intuitions about the nature of the data. He decides what constitutes similarity in a given case.

2.3.1 Preliminaries

Via an appropriate kernel, the information given in a set of even "interestingly structured" data (in their 'questions for future research', Jäkel et al. (2009) explicitly mention strings and trees as representations for sentences) is reduced to a kernel matrix. This matrix is the "central data type of all kernel-based algorithms" (Shawe-Taylor and Cristianini 2004). A *kernel method* is an algorithm operating on the kernel matrix. The general structure is given in 2.25, adapted from fig. 2.4 in Shawe-Taylor and Cristianini (2004).

(2.25) Overview of kernel methods



The kernel matrix is a bottleneck in the flow of information. The two sides of the informational bottleneck are independent of each other: any kernel can be combined with any kernel method (Jäkel et al. 2009, Shawe-Taylor and Cristianini 2004). Hence kernel methods are *modular*. They keep apart the data analysis method and the characterization of similarity.

In general, kernel methods are learning algorithms for detecting linear patterns in the feature space. The feature extracting function ϕ may extract non-linear features. For example, ϕ could extract the square value of some measurable surface feature. Consequently the detected patterns do not need to be linear in the original space. They are linear only in the space of extracted features.

Let $\phi_j(x)$ extract the jth feature of a data item $x \in R$. A *perceptron* defines a linear decision boundary in feature space, given by weights w_j ranging over the training set. These weights are trained or computed to optimally divide the data set into members and non-members of a class. x is a member of a class C if the linear combination of its features (their weighted sum) exceeds a threshold b (cf. Jäkel et al. 2009):

(2.26) A perceptron:

$$p(x) := \sum_{j=1}^{m} w_j \phi_j(x)$$

$$p(x) \geq b \rightarrow x \in C$$

According to the *representer theorem* (Schölkopf and Smola 2002), the optimal weights w_j can be expressed as a linear combination of features of training items $x_i \in A$. The weight w_j is the sum of the values of the jth feature, weighted by a weighting vector u, over all training data.

(2.27)
$$w_j = \sum_{i=1}^{n} u_i \phi_j(x_i)$$

Combining the definition with the representer theorem, the perceptron can be re-formulated in terms of products of features, and consequently in terms of a kernel k defined by the feature mapping ϕ (Jäkel et al. 2009):

(2.28)
$$p(x) = \sum_{i=1}^{n} u_i \sum_{j=1}^{m} \phi_j(x_i)\phi_j(x) = \sum_{i=1}^{n} u_i k(x, x_i) = u^T \overrightarrow{k}(x)$$

The perceptron now derives class membership via a weighting u of the kernel vector for x. This model both instantiates a decision boundary and is exemplar-based, as argued by Jäkel et al. (2009). Kernel-based decision boundary models are discussed in section 2.3.2.

2.3.1.1 Factor analysis

The following sections discuss various instances of *factor analyses*. Here I give a short overview of their general architecture.

Both in explicit feature models and in kernel models, the data spans a vector space. Kernel methods are operations in the m-dimensional feature space, where each feature is a dimension. By the implicit feature mapping ϕ, each data item is mapped to a vector in the feature space. One can extend the data set to the set of linear combinations of data vectors, the data space, expecting every point in this space to be a *possible* data item.[10] If centered, n data items can only span a $n-1$-dimensional space.

In many conceivable applications, the number of features m exceeds n. Then, the data space is a subspace of the feature space. This means that not all points in feature space are also points in the data space, and thus there are unattested combinations of features.

It is a step of abstraction to identify the subspace of the feature space in which the data lie. If less than $n-1$ dimensions suffice, the data was redundant. Then the true variation of the data and possibly a more concise description of each data point is found with a low-dimensional basis for the data subspace.

Factor analyses (such as *Latent Semantic Analysis* (2.3.3.1), *Independent Component Analysis* (2.3.5) and *Principal Component Analysis* (2.3.4)) derive a basis for this data subspace, which is orthogonal in terms of items as well as in terms of the implicit feature vectors. The basis vectors can be given as (not necessarily unique) linear combinations of data items, establishing orthogonality by

[10] Respectively, their pre-images; see section 2.4.4.

items, and they can be extended to (unique) linear combinations of features (via multiplication with the feature vector for each item, given by ϕ). This establishes orthogonality by features.

If features are taken to be implicit only, the basis vectors need to be given in terms of data items. A point in the data space can then be given as a vector of coordinates in this basis, or, equivalently, as a linear combination of data items (possibly including itself, and not necessarily uniquely). The basis vectors are sometimes called the 'hidden features' of the data set. I will abstain from using this term in this sense, since they are unlike surface features or what I call the hidden features (the grammatical features posited by the linguist); they are neither empirical nor theoretical constructs, they are merely mathematically derived from the data set. Yet by the very name factor analyses are meant to derive *analyzable, interpretable* factors. This issue is further discussed in section 2.3.4.3.

Again assuming implicit surface features, new data items need to be projected into the data subspace according to their kernel vector because no other information is accessible (shown for Kernel PCA in section 2.4.1 and in appendix B). If a new item is linearly independent from the training data some information (such as previously unseen features or unseen combinations of features) will be lost in the projection, because its actual image in the feature space lies outside the subspace. We then only get an approximation of the actual image. This can also be a benefit, because it may abstract over noise and other artifacts in the data.

Data items can thus be represented in several different ways: First, by an explicit feature vector. Second, by their kernel vector, that is, as a vector of similarity values. This is a very indirect characterization of a data item. Thirdly, they are represented as linear combinations of the training data. This is much more direct, almost like a building instruction. Each known data item is paired with a coefficient: Take this amount of this item, subtract that amount of that item, and so on.

This characterization attempts to model a data item in terms of the kernel's implicit features. Yet *all* features are included in this construction process, also the ones which are not included in ϕ, but only in the extractor of *hidden* features ψ. Technically ψ does not differ from ϕ. ϕ lists observable features, ψ lists additional features posited by the linguist. It is his choice which features to declare observable and which not, and this choice depends on the task. The kernel is defined by ϕ, so the features in ψ are not available to it. By this definition, kernel values cannot take into account the similarity of hidden features, only that of surface features.

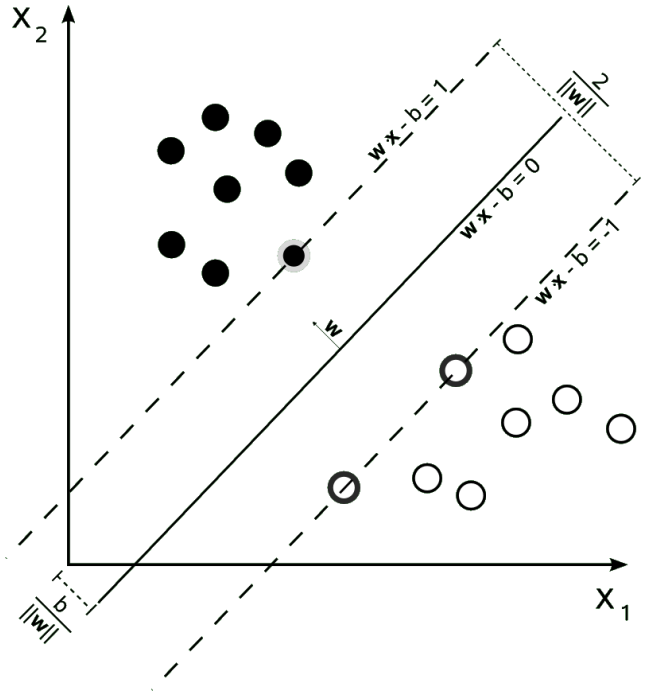
From the description as a linear combination of the data the new item now also receives features from ψ, according to their respective coefficients. This is indirect, because the description is only based on the similarities according to ϕ. Yet it allows *inferring* the hidden features, founded in surface similarity. Similar to an exemplar model, where a new item inherits its properties from the items it is most similar to, in factor analyses it inherits them from those items needed to construct it.

2.3.2 Support Vector Machines

By far the most common application of kernels are instances of **Support Vector Machines** (SVMs). Cristianini and Shawe-Taylor (2000) give an introduction and overview. SVMs find a hyperplane separating the feature space into two classes of data items. This hyperplane is defined by a vector w

which is perpendicular to it. It is thus given as the set of vectors x satisfying $w \cdot x - b = 0$, where w defines the orientation of the hyperplane and $\frac{b}{|w|}$ is its distance to the origin. The decision boundary and the margins are shown in fig. 2.29.

(2.29) Support Vectors[11]



Parallel to the decision boundary are the margins $w \cdot x - b = 1$ (positive margin) and $w \cdot x - b = -1$ (negative margin). w and b are chosen to maximize the margin $\frac{1}{|w|}$. Then $w \cdot x - b \geq 1$ for all x belonging into the one class, and and $w \cdot x - b \leq -1$ for the others. The *support vectors* are the data items lying on the margins. As a decision boundary model, SVMs are an extension to perceptrons.

Of course such a solution can only be achieved if the data is in fact linearly separable. A *soft margin SVM* (Cortes and Vapnik 1995) finds a hyperplane that minimizes how far data items lie on the wrong side of the respective (positive or negative) margin.

Non-kernelized SVMs operate on explicit feature vectors x. The first advantage of kernels now is that the hyperplane does not need to be linear in the initial feature space, because a kernel can be defined as a (non-linear) function over these initial features. If an appropriate kernel can be defined, it induces a feature space where the data can be separated linearly.[12] Generally, non-linearity is preferred to be put in the kernel as opposed to the learning algorithm, since linear learning algorithms are much more efficient.

The second advantage is that non-vectorial data can be treated as well, by defining a kernel over them. In both cases, place data items x by their features $\phi(x)$. The inner products are now replaced by kernel computations.

[11] `http://en.wikipedia.org/wiki/File:Svm_max_sep_hyperplane_with_margin.png`, with a correction by P. Buch.

[12] This is trivial if the kernel has access to class memberships. Let $\phi(x)$ be $\langle 1 \rangle$ if x belongs to the class, and $\langle 0 \rangle$ otherwise. The data items fall onto two points (1 and 0) in the 1-dimensional feature space, where they can be separated linearly.

2.3.2.1 Applications in linguistics

As classification tasks are abundant, so are SVM approaches. I can only name a few examples.

Text categorization. Documents are characterized by the words they contain (bag-of-lemmas kernel, Joachims 1998) or viewed as word sequences, captured by string kernels (Lodhi et al. 2002).

Relation extraction. Relation extraction is a very active field of research, and it is being tried with lots of different kernels, see Zelenko et al. (2003), Culotta and Sorensen (2004), Bunescu and Mooney (2005a,b) and many more. Syntactic relations between words (given by a tree fragment, a dependency path, or simple co-occurrence at a certain distance) are mapped to semantic relations.

Semantic role labeling. Moschitti et al. (2006) define a kernel on syntactic structures relating predicates to their arguments in order to classify the latter by semantic role.

Co-reference resolution. Versley et al. (2008) employ different kernels over the syntactic environment of pairs of NPs to predict (by SVMs) whether they are co-referent, that is, whether they denote the same entity; and for instances of the pronoun "it", whether it is expletive or not.

Morphological classes. Nastase and Popescu (2009) use SVMs to separate German plural classes. A detailed discussion is given later when I compare it to a KPCA model of German plural (3.3.3).

Parse ranking. In a series of papers (2001a, 2001b, 2002), Collins and Duffy demonstrate the use of kernels over syntactic trees or descriptions from Lexical-Functional Grammar (LFG) to rank a set of candidate parses for a given sentence. Training data are pairs of sentences and correct or incorrect parses; a perceptron (closely related to SVMs) then identifies a boundary (hyperplane in the feature space) separating the two classes. This model explicitly learns from negative data, having two classes (grammatical vs. ungrammatical). Candidates are ranked according to their distance to the separating hyperplane. Collins and Duffy extend the model to part-of-speech tagging and named entity recognition, with appropriately defined kernels. Their model is further discussed in section 7.1.

Semantic parsing. Kate (2007) defines a set of semantic relations and trains an SVM for each to classify portions of sentences. An Earley-style chart parser then derives a semantic structure (for a natural language query).

2.3.2.2 Conclusion

Decision margins (as in SVMs) describe classes in terms of their boundaries. Classification is done by calculating on which side of the boundary an item resides. This resembles rule-based approaches more than exemplar models. In exemplar models, classes (and other abstract features) are determined

by an item's closeness to others, not by its distance to a derived construct (the boundary). These two views might lead to the very same classifications — both are mathematical abstractions over classification. Nosofsky and Stanton (2005) conclude in their experiments that exemplar models better predict human behavior than decision boundaries.

2.3.3 Distributional Semantic Models

Distributional Semantic Models characterize the meaning of a word by the contexts it appears in. Contexts for semantic classification usually are documents (and not only sentences), so meaning is rendered as topic-relatedness. Words appearing in the same contexts are conceived as conceptually similar.

The basic data type of such models is a document-by-term occurrence matrix: the entries denote how often each term (a word, excluding a list of stop words) occurs in each document. Thus every term is encoded as a vector over documents, and likewise every document is encoded as a vector over terms (the so-called *bag of words* representation). The representations for terms (or documents) can be compared to each other, resulting in pairwise distances or similarities, which in turn can be used in clustering etc.

A recent discussion of DSMs is given by Turney and Pantel (2010). These models are used to find synonyms (Landauer and Dumais 1997), to predict the strength of priming between two words (Lund and Burgess 1996), or to classify nouns into a pre-defined set of semantic classes (Levy et al. 1998).

On the syntactic side, the environment of a word is not a whole document (to which it is supposed to be topic-related), but a window of surrounding words (or similar local features). Levy et al. (1998) conducted a second experiment in which they classify words into 12 part-of-speech classes.

2.3.3.1 Latent Semantic Analysis

Factor analyses calculate a basis of the feature space, that is, a set of (in most variants: orthogonal) vectors. Representing a data item relative to this basis has advantages over the original feature vector (here: occurrences over documents): the representation is usually shorter, and the factors are uncorrelated.

The original surface features denote occurrences in documents. The ith feature thus reads as "probably related to the topic of document i", which is not a meaningful semantic characterization. The extracted, orthogonal vectors are intended to be more meaningful. They are interpreted as the *latent features* (not to be confused with the hidden features; in the terminology used here, hidden features are the ones stipulated by the researcher) of the data; however I argue that this notion of features is not very meaningful in linguistics (2.3.4.3). Nevertheless, it is useful for characterizing data items by just a few prominent properties, as employed in Latent Semantic Analysis and related approaches.

Latent Semantic Analysis (LSA; Deerwester et al. 1990) is a Singular Value Decomposition (SVD) of a word-by-document occurrence matrix A. Compared to the previously mentioned explicit

feature matrices (m features by n data items), this matrix has the documents as items (columns), and the features are counts how often a certain word occurs in it. Conversely, the transpose of A has words as data items, and they are characterized by the documents they occur in.

A is first centered by subtracting the mean value of each feature, and then decomposed via SVD as $A = U\Sigma V^T$, where U (V) are the $m \times m$ left (resp. $n \times n$ right) singular vectors, and Σ is a $m \times n$ matrix with the non-negative *singular values* on the diagonal. The singular vectors serve a twofold purpose: V rotates the word representations A onto its latent dimensions, which are given in U (times the singular values); and conversely U rotates the document representations onto V:

(2.30) Word representations: $A' = AV = U\Sigma V^T V = U\Sigma$

Document representations: $(A')^T = A^T U = (U\Sigma V^T)^T U = V\Sigma U^T U = V\Sigma$

The Euclidean distance or the cosine of the angle between two feature vectors (both usually work) is taken as a measure of distance. Thereby one can cluster words by meaning. Clustering is a useful application independent of the actual interpretations of the latent dimensions. Some latent dimensions turn out as highly correlated with some semantic concepts. However, the dimensions are more of a practical value in the clustering of words than in the decomposition of meaning. For this purpose, one only looks at the few most informative dimensions. Most informative are those with highest singular values. This reduction of dimension further reduces the number of features needed to represent the distributional properties of a term. Distributional properties are taken to be indicative of semantic properties in distributional semantics. In that sense LSA extracts the meaning of words.

2.3.3.2 Kernel LSA

Cristianini et al. (2002) reformulate LSA to work on a document-by-document matrix, obtained by multiplying the document-by-term matrix by its inverse. If a document's explicit features are the terms occurring in it, then the cells of this matrix are vector products of document representations. Now this is the definition of a kernel, and thus the matrix is a kernel matrix. Essentially, this method then is kernel LSA. Cristianini et al. (2002) demonstrate it for text classification.

2.3.3.3 Hyperspace Analogue to Language

Lund and Burgess (1996) define another instance of DSM, *Hyperspace Analogue to Language* (HAL). The initial feature representation of a word counts other words preceding or succeeding within a certain, narrow window (of 5 words). Out of these, the 200 most variant ones are chosen, because they are most informative. Preceding and succeeding occurrences are counted separately, so there are 400 features in the final surface representation. Lund and Burgess's definition of distance is Euclidean, or — more generally — the Minkowski r-metric.

Local lexical co-occurrences reflect both syntactic and semantic restrictions. In theory, the latent dimensions (obtained via a factor analysis) should denote such restrictions, but in practice interpre-

tation is much harder. Therefore HAL also better serves for purposes of visualization (by projecting the data into a 2-dimensional space) or clustering than of interpretation.

2.3.4 Principal Component Analysis

Principal Component Analysis (PCA) (PCA; Jolliffe 1986/2002) is a very common form of factor analysis. In a multi-dimensional feature space, it identifies a direction along which the data maximally varies. PCA calls this direction with maximum variance the 'first principal component' of the data set, and removes this variance. This is achieved by a projection, which can be thought of as flattening the cloud of data points along this dimension. In the remaining subspace, all vectors are orthogonal to the one just projected away. Out of these, PCA again finds the one with maximal (remaining) variance: the second principal component. This process is continued until there is no variation left in the data set. This leads to a unique[13] solution for each data set. The principal components thus form an orthogonal basis of the data space, and they are ordered by the amount of variance they describe.

2.3.4.1 Definition

A data matrix A, where each of the n columns denotes a data item and each of the m rows denotes a feature, is first centered. This is done by subtracting the mean feature vector from every data item. In the following, I will denote the centered matrix with entries $A_{i,j} - \frac{1}{n}\sum_{k=1}^{n} A_{i,k}$ by A again. Then $\forall i \leq m\colon \sum_{j=1}^{n} A_{i,j} = 0$.

In feature space, this is a translation of all data items. Centering is important because it causes the origin of the basis to be calculated to lie within the data space. Beforehand the center was the null feature vector, which in most conceivable applications is not a description of a valid data item. Thus the center lies outside the subspace of the data within the feature space. If that is the case, at least one dimension in feature space which is perpendicular to the data subspace is needed, pointing from the origin into the space. This dimension describes variation which is not present in the data. Every data item would need to have the same value along this dimension, and thus it is meaningless. Other values describe points outside the data space.

After centering, the feature co-occurrence matrix AA^T is decomposed as $AA^T = U\Delta U^T$, where U contains the principal components and Δ is a diagonal matrix with the (non-negative real) eigenvalues on the diagonal, in descending order. Let r be the number of non-zero eigenvalues, U_r the matrix containing only the principal components corresponding to those, and Δ_r accordingly.

The decomposition is done via SVD: $A = U\Sigma V^T \rightarrow AA^T = U\Sigma V^T(U\Sigma V^T)^T = U\Sigma\Sigma^T U^T$. The eigenvalues are the (point-wise) squares of the singular values: $\Delta = \Sigma^2$. Hence there is an immediate connection between PCA and LSA. Each column of U defines a *principal component* as a linear combination of surface features. The surface representations of data items can thus be rotated onto the factors via $A' := U^T A$, just as in LSA. This is a change of basis, from features to

[13] Unique up to arbitrary rotations of directions with identical variance.

factors.

2.3.4.2 Reducing the dimension

The ordering of principal components can be exploited to get a smaller model, by discarding low-ranked components. To do so, U_r is replaced by the l principal components with highest eigenvalue U_l, for some $l < r$. Each data item is then described in l factors. With only a fraction of the original components some models still achieve nearly the same accuracy as the full model (for an example, see 4.1.1).

Knowing a data item's position on more dimensions pins it down more precisely, with smaller gains from each following component. At some point, the explanatory value drops below that of a single exemplar. This is a reasonable point to cut off. Principal components below this point are merely cosmetic. They tweak a data item's representation up to a 100% fit, which often is not needed.

If we assume noisy data, a full fit is not even wanted. We expect the higher components to capture the real variation in the data, and the lower ones to account for imprecision of measurement and other noise. PCA has been successfully applied to restoring data sets where random noise was added.

A linguistic model with as many dimensions as there are data items has little explanatory value. The aim is a model as small as possible, and as large as necessary to capture the data.

2.3.4.3 Interpretation of the principal components

It is appealing to get the *inherent dimensions of variation* within a data set, using PCA. Yet it is important to be aware of what these dimensions mean. Even more important is to know what they cannot be.

Along a dimension of maximal variance, the data is maximally spread. It is by no means necessary that there are dense clusters of items at either side (although such a case would point PCA to establishing this dimension as a principal component). PCA does not identify clusters, and neither will it posit or prove the existence of abstract classes.

Principal components are sensitive to frequencies. Therefore they heavily depend on the data and cannot denote immutable categorical properties of (linguistic) data. Every new data item will shift the components a little bit. This might be a reasonable assumption in a cognitive model (every experience changes our mental representations), but it does not yield scientific abstractions.

Data items receive a representation as a vector over principal components, yet the interpretation that a data item is *composed* of them (as a sort of building blocks) is misleading. All components are needed, so it is not the case that ontologically 'smaller' items (shorter words etc.) contain less 'components', and the components are needed to various degrees. This continuity of possible values is incompatible with an atomic interpretation. Also, no one would claim that a linguistic item is composed of as many parts as the linguistic space it resides in has dimensions.

The uninterpretability as claimed here only applies to principal components, not to dimensions of the (linguistic) spaces in general. The space as a whole is used successfully in every application in this thesis. Other factors can be obtained, for example by rotating the existing ones (see 4.2). Factor rotations (section 4.2) will yield more useful dimensions, yet a 'building block' interpretation will be almost impossible to obtain, as I will argue for in 4.5: Even for a minimalistic example with pre-defined building blocks, these can only be approximated.

In conclusion, there is no theoretical linguistic interpretation to principal components. The maximum of variance in the data is not a linguistic category. There might be applications in linguistics where variance plays a role, e.g. in phonetics, but this is not the case in traditional phonology, morphology, syntax and semantics. Maybe this instantiates a more general problem: that data-orientation is in fact necessary, and that abstract categories of language never become unambiguously manifest on the surface and therefore can only be posited in an abstract way. Any change in frequency in the data will immediately distort the established categories. Data-oriented models acknowledge that the mental representations of language contain no abstractions.

2.3.4.4 Applications in linguistics

There are only a few applications of Principal Component Analysis in linguistics. I know of one with kernels (see 2.4.7) and two without.

Lexical typology. Jäger (2010, to appear) classifies color terms cross-linguistically using PCA on explicit vectors representing the area in the physical space of colors which a certain color term denotes in a certain language/idiolect. The color space is given in the form of the 330 Munsell color chips. Native speakers of unwritten languages all around the world were asked to name these colors (World Color Survey, Cook et al. 2005).

For 1771 speakers (from 102 languages) the data set is complete for all 330 chips. They used a total number of 1601 color terms. These are the data points. Each has 330 features: The number of speakers who used this term for that color.

Arguably, to know the denotation of a color term, one does not need to know 330 features. Jäger uses PCA to extract the most relevant factors, and finds that the first 15 of them explain 91.6% of the variation across color terms. Discarding the others hopefully removes annotation errors, but also non-significant inter-speaker variation, and retains the universally significant tendencies.

Across languages, a distinction between red/yellow and green/blue is most dominant. It is in the nature of principal components to be twofold partitions of the feature space; they do not denote single colors. In order to obtain interpretable results, Jäger (to appear) applied a factor rotation (VARIMAX, Kaiser 1958; see 4.2.2) which maximizes the correlation of the factors with the surface features. The resulting 15 dimensions denote 10 of the 11 'universal colors' (except gray, which was underrepresented in the data), and 5 additional shades.

PCA can thus be used to identify cross-linguistically stable semantic concepts as subspaces of the semantic space (here: subareas of the color space).

Visualization of linguistic spaces. Plunkett and Nakisa (1997) visualize the distribution of Arabic plural classes in the space of singular nouns in two dimensions. These dimensions are the first two principal components of a PCA on explicit surface feature vectors of singular nouns. This representation captures as much variance in the data as possible. Nouns fall into clusters corresponding to the number of syllables, and there appears to be some correlation between a noun's position in this two-dimensional space and its plural class.

KPCA was not available back then, otherwise a kernel would have been a better choice than the explicit feature vector. However, they continue to predict fully specified plural forms with an artificial neural network, and therefore they need explicit input features. Their paper is not only noteworthy for the use of PCA, but also for a connectionist method for predicting output forms (see section 2.1.4). It is thus the direct predecessor of my KPCA models of inflection in chapter 6, where I combine the ideas in employing KPCA directly to predict plural forms instead of just their class.

2.3.5 Independent Component Analysis

Independent Component Analysis (ICA; Hyvärinen et al. 2001) is a factor analysis similar to LSA. There is a kernelized variant (Bach and Jordan 2003), but I am not aware of linguistic applications. So in this sketch I describe standard ICA, following Honkela et al. (2010).

Remember that LSA is an SVD of a features-by-words matrix: $A = U\Sigma V^T$. U contains the *latent dimensions* of the semantic space. These form a basis of this space, but as also argued by Honkela et al., p. 286, they are hard to interpret. Another basis for the same space could be more meaningful, and it can be obtained by rotating the vectors (cf. sections 2.4.5.1 and 4.2) in U: $U = RW$, where R is a rotation matrix and W contains *independent factors*. ICA performs this decomposition of U according to a criterion of *independence* of the factors.[14] The vectors in W span the same space as the latent semantic dimensions, but by the independence criterion they are supposed to be more interpretable.

As in LSA, the number of latent dimensions can be reduced by retaining only the ones with highest singular value. This vastly reduces computation time. A similar point will be made in 4.2. Instead of SVD, PCA can be used to obtain a set of initial factors. In any case, ICA needs orthogonal factors as an input. This is generally not given, so either SVD or PCA is needed. Additionally, these methods allow for reducing the dimension.

2.3.5.1 Applications in linguistics

ICA is used by Honkela et al. (2010) to obtain multi-dimensional, semantic representations for words from the environments they occur in. The feature matrix A contains explicit feature vectors for each word. Features are co-occurrence counts for 200 other words, separated into preceding and succeeding occurrences. This yields 400 features in total. Only occurrences within a fixed window

[14] This criterion is the *kurtosis* of the factors, not the *variance* as in SVD or PCA; a similar, yet higher order measure. Kurtosis is the 'peakedness' of a distribution, with a Gaussian having a kurtosis of 0. According to Calderone (2008), ICA maximizes the 'non-gaussianity' of the factors. Refer to Hyvärinen et al. (2001) for details.

size (e.g. 5) count, and closer distances count more (e.g., 1 for a distance of 5 linearly increasing to 5 for a distance of 1).

The 200 words are chosen to be the ones with maximal variation; that is, the ones which are most informative. This preparation step is an initial reduction of dimension, but it also exemplifies the constraints imposed by explicit feature vectors.

Taking the nearer vicinity of a word as features leads to a representation which does not capture topic-relatedness (as in LSA, where all other words in the document are features), but local distributional patterns. These patterns roughly correspond to word classes (part-of-speech), and to selectional preferences between heads and arguments. The feature space is thus not a semantic (or topic) space, but more of a syntactic space.

Data points (words) can be represented using a small number of independent components. Honkela et al. visualize the coordinate a word receives in the space spanned by these components using bar plots. Presumably similar words are represented similarly, so they are close in the syntactic-semantic space. There is a component correlating with being an adjective, and one for being a verb; such that adjectives and verbs take a high value on the respective dimension. Similarly, Honkela and Hyvärinen (2004) also only give tentative results.

Calderone (2008) identifies linguistically meaningful factors in Italian verb forms, using a sample of 51 verb forms. Each letter is a data item, and its surrounding letters (a window of 5 to the left and 5 to the right, including a special symbol for positions lying outside the word) are encoded as its features in a binary vector. These features are first condensed to 50 principal components (using PCA). ICA then rotates these factors. The resulting factors are manually examined and visualized to show that they (or at least some of them) are highly correlated with either a verbal stem (that is, words derived from that stem receive a high absolute value on this dimension) or a morphological feature, such as ‘first person plural’.

This correlation is mediated by the surface features: The factor denotes a set of surface features, which appear in a certain morpheme, which in turn is the exponent of a grammatical feature. It is important to notice that these grammatical features were not available to the method; in a sense, those morphemes were discovered. This is due to the concatenative nature of the data set: A sample of stems combined with suffixes. ICA seems to be able to extract a separate dimension for each stem and each morpheme.

Lagus et al. (2004) characterize Finnish morphemes by their distribution. A corpus of Finnish words is semi-automatically parsed into morphemes; the 3759 most frequent ones act as the data items. The 505 most common words plus an end-of-word boundary are the features. For each data item, it is counted how often it is immediately followed by each of them. This gives a 506-dimensional initial representation for each data point, which is again reduced to 50 principal components. After applying ICA, the rotated factors reportedly are correlated with certain syntactic, semantic or morpho-phonological properties. An example of the latter would be frontness of vowels, which due to vowel harmony sorts morphemes into two classes; however, this is only tentatively given as an interpretation. Two factors are interpreted by visual inspection.

2.3.5.2 Conclusion

While this review is not complete and Väyrynen and Honkela (2005) conclude that ICA is superior to unrotated SVD (that is, classical LSA), the implementations discussed here only convey tentative results and visual inspections. ICA acknowledges the problems in interpreting latent semantic dimensions, and offers rotation as a solution to obtain another, more meaningful basis of the semantic space. I will come back to rotations in 4.2. The studies cited here restrict the context, so that also syntactic and morphological features play a role. This can be thought of some sort of latent *syntactic* analysis.

2.3.6 Conclusion

Many learning algorithms have been re-formulated in terms of inner products of feature vectors. The resulting kernel methods are state of the art in machine learning. Kernels make features implicit, which allows for more flexibility, up to a virtually unlimited number of features. This would be unfeasible in neural networks and the like.

Connectionist methods integrate the similarity measure into the model. Kernel methods separate the two and thus are modular. Any knowledge about the domain can be hidden from the learning algorithm. This is superior to designing the internal layers of an artificial neural network.

At the same time, kernel methods are closely related to exemplar models (Ashby and Alfonso-Reese 1995, Jäkel et al. 2008): They operate on the kernel vector, the vector of pairwise similarities to all stored exemplars. Therefore kernel methods are a valid mathematical abstraction over such cognitive models.

2.4 Kernel Principal Component Analysis

What makes PCA especially interesting is its reformulation for kernels (Kernel PCA / KPCA, Schölkopf and Smola 2002, Schölkopf et al. 1997, Shawe-Taylor and Cristianini 2004), because explicit feature vectors are no longer necessary. The kernel implicitly defines a feature space: Each data item is implicitly given as a vector of its features, and this vector is its coordinate in a space where each feature is a dimension. Now usually we have far more features than data items, and to tell apart n data items, at most $n - 1$ dimensions are needed. The data items fall into a subspace of the feature space. Every point in it can be expressed as a linear combination of data items. KPCA calculates a basis for this subspace.

KPCA is very popular in machine learning, yet somewhat under-represented in linguistics. The only linguistic application (Clark et al. 2006, Costa Florêncio 2007) is discussed in 2.4.7. I conclude that there is a large gap between kernelized factor analyses and linguistics; and I intend to lay some groundwork here. In section 2.4.2, I discuss the incorporation of frequencies into KPCA. To this end, I discuss *Weighted Kernel PCA* (Wang et al. 2005), which fulfills the requirement of being equivalent to a standard KPCA on an artificially extended data set simulating frequencies, where

identical items occur several times. This will be proven in appendix C.

KPCA will be the central method of this thesis. This choice is mainly due to its popularity in ML — PCA is called the "perhaps [...] most common feature extraction algorithm" by Schölkopf and Smola (2002, p. 19) — and its accessibility to the non-expert[15], among kernelized factor analyses.

2.4.1 Definition

Here I only sketch KPCA; the full definitions are found in appendix B.

Let A be the data matrix with n columns containing the implicit feature vectors $\phi(a_i)$ for each data item a_i ($0 < i \leq n$). Then $A^T A = K$, the kernel matrix, because its entries are inner products of the feature vectors. In KPCA, the kernel matrix is centered, such that each row (and by symmetry, each column) adds up to 0. Then it is decomposed as $Q\Delta Q^T$, where Δ is the diagonal matrix of eigenvalues in descending order, and Q are the corresponding eigenvectors. Define α as the eigenvectors normalized by the inverse square roots of their respective eigenvalues. The principal components are then given as linear combinations of A: $U = A\alpha := AQ\Delta^{-\frac{1}{2}}$.

Let r be the number of non-zero eigenvalues. Then U_r are the r principal components. It consists of implicit feature vectors, but these are never used explicitly. The r new dimensions of a data item z are given as $\alpha^T \left(\overrightarrow{k}(\overrightarrow{a}, z) - Kv \right)$, where v is an averaging vector (a column vector with values $\frac{1}{n}$ everywhere) and $\overrightarrow{k}(\overrightarrow{a}, z)$ is the kernel vector for z. Left multiplication with α yields the coefficients ζ of a linear combination of data items. This linear combination is a description of z in terms of the data set. From this representation, the hidden features of z may be inferred (2.4.3). For example applications, see 2.4.6 and 3.1.

2.4.2 Weighted Kernel Principal Component Analysis

A main purpose of many machine learning techniques in linguistics is to incorporate frequency data into the model, in order to explain or model effects of frequency in natural language. In factor analyses, it is desirable to have frequency-sensitive factors, which shift towards heavier clusters of items. It does not matter whether this cluster consists of many highly similar items or of numerous instances of one and the same type: The effect shall be the same.

2.4.2.1 Doing it explicitly

One way to mimic this behavior in KPCA is to multiply each row and column of the kernel matrix by the frequency of the respective data item. Let this kernel matrix be the *expanded kernel matrix*. It is computed via an expansion matrix P, with item frequencies f_1 to f_n, the sum of which is $s := \sum_{i=1}^{n} f_i$.

[15] Which I, after writing this thesis, still consider myself.

(2.31)

$$P = \underbrace{\begin{bmatrix} \underbrace{1\cdots1}_{f_1} & 0\cdots & \cdots & \cdots 0 \\ 0\cdots0 & \underbrace{1\cdots1}_{f_2} & & \vdots \\ \vdots & & \ddots & \vdots \\ 0\cdots & \cdots & \cdots 0 & \underbrace{1\cdots1}_{f_n} \end{bmatrix}}_{s}$$

P duplicates rows/columns: left multiplication with the transpose of P and right multiplication with P takes the n-by-n kernel matrix to its expanded (s-by-s) version, just as if each data item had been in the data set several times, according to its frequency. The expanded kernel matrix now contains many duplicate entries. It is obviously much larger than the original matrix. It is centered and decomposed as in ordinary KPCA. Consequently, storage and computation will be more costly.[16]

The principal components necessarily contain identical coefficients for identical items. They are expressed in terms of the expanded data set. In order to interpret them with respect to the original data set, they are summed up item by item. This is done via left multiplication with P. The principal components thus derived shall be the standard against which to evaluate the alternatives.

2.4.2.2 Reformulation

Alternatively, it is possible to include the appropriate frequency factors into KPCA in some way such that the resulting principal components (and eigenvalues) are the same. This technique is called *Weighted Kernel Principal Component Analysis*, or WKPCA, for short (Wang et al. 2005). As usual, the definitions — and my proof of equivalence of WKPCA and KPCA on the expanded kernel matrix — are in appendix C. Here I will give a brief introduction and the intuitions behind WKPCA.

The most reasonable place to apply frequencies is in the averaging factor, W (such that the weighted and centered matrix M_W equals WKW^T).

(2.32)

$$W := \left(I_{n,n} - \frac{1}{s} f^{\frac{1}{2}} {f^{\frac{1}{2}}}^T\right) diag(f^{\frac{1}{2}})$$

$$WKW^T = \left(I_{n,n} - \frac{1}{s} f^{\frac{1}{2}} {f^{\frac{1}{2}}}^T\right) diag(f^{\frac{1}{2}}) K diag(f^{\frac{1}{2}}) \left(I_{n,n} - \frac{1}{s} f^{\frac{1}{2}} {f^{\frac{1}{2}}}^T\right)$$

Note that W is n-by-n and not s-by-s, although of course $\frac{1}{s}$ is used for averaging. In a first step, the kernel matrix is weighted by multiplying it with the square root of the frequencies both from the left and from the right. The remainder of W then is symmetric (W is not): $\left(I_{n,n} - \frac{1}{s} f^{\frac{1}{2}} {f^{\frac{1}{2}}}^T\right) =$

[16] An already computation-heavy procedure such as eigenvalue decomposition cannot afford this while still being practical.

$\left(I_{n,n} - \frac{1}{s} f^{\frac{1}{2}} f^{\frac{1}{2}T}\right)^T$. This part is similar to KPCA's centering matrix V. It contains of the identity, and subtracts not a uniform average, but an average weighted by (the pairwise products of the square roots of the) frequencies.

W is not a true centering factor. It is only a substitute for the centering factor of the expanded kernel matrix. Consequently the kernel matrix M appears not to be centered. The sums of each row (or column) are not 0, because this summing implicitly assumes equal weights. However M is centered with respect to the weighting.

In deriving principal components equivalent to an expanded KPCA as described above, α needs to be adjusted as follows.

(2.33)
$$\alpha_{Wr} := W^T Q_W \Delta_r^{-\frac{1}{2}}$$

Note that if the equivalent is done in standard KPCA (multiplication with V^T) its principal components remain unchanged since they are already centered: $\alpha = Q\Delta^{-\frac{1}{2}} = V^T Q \Delta^{-\frac{1}{2}}$. For the same reason as with W above, the principal components of WKPCA are — after multiplication with W^T — not orthogonal with respect to the data *types*; they are orthogonal with respect to the data *tokens*. This can be seen from the fact that the same principal components are derived from the expanded kernel matrix, which shall be proven in the appendix (appendix C, lemma 13). This proof of equivalence then justifies the definitions of W and α_W.

Computationally, WKPCA is only slightly more expensive (one more matrix multiplication) than unweighted KPCA. Compared to an expanded KPCA, it saves both time and space, while arriving at the same solution. It is therefore a valid technique to incorporate frequencies into KPCA. Frequencies do not have to be integers; I expect WKPCA to be well-behaved on the positive reals. I will not explore the effects of negative frequencies here.

WKPCA is exemplarily applied in section 3.2.3 to a KPCA model of gender assignment in German, and in section 4.1 in conjunction with dimension reduction. The general hypothesis is that weighted data result in better performance, because they more naturally model the native speaker's experience with language.

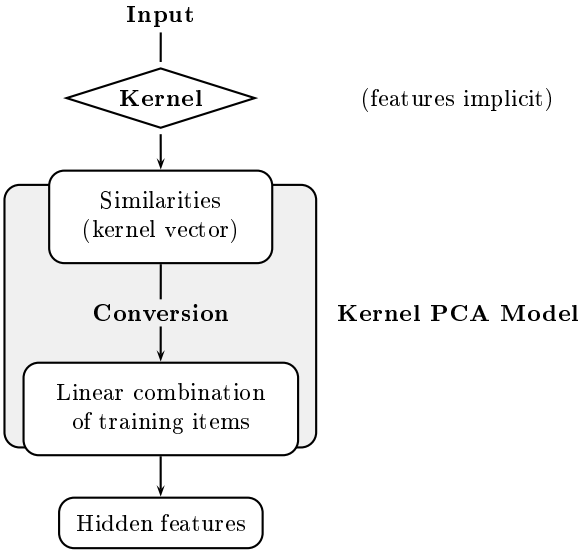
2.4.3 Feature inference

Given a set of linguistic objects (training set) and a kernel over their *surface* features, KPCA calculates the feature space in which all these objects lie. A test item is represented as a linear combination of the training items. Its features are a linear combination of the features of the data items. Now assume that the hidden features are correlated to the surface features. As the linear combination is constructed to best approach a test item, I interpret it also as an attempt to reconstruct its hidden features. Then not only the surface features can be read off, but also the hidden features.

Given a hidden-features-by-items matrix B, the hidden features of a linear combination with coefficients ζ are given by $B\zeta$. There is thus is a direct and exact computation of the inferred

features (see lemma 9 in appendix B). In network models, a weighting has to be *trained* from the layer of activation strengths of words to the output layer.

(2.34) Architecture of feature inference with KPCA



In Naive discriminative learning (Baayen et al. 2011), the input is the explicit surface feature vector. Using the kernel vector skips over the layer of surface features.

The KPCA inference model relates to the explicit model (2.9) via their input vectors. Both are vectors over the training set. In the explicit model, it is an activation vector depending on the pairwise similarities: The more similar, the higher the activation. If activation spreads via (unweighted[17]) surface features, then each activation equals the sum of features shared by a word and the input. This computation is already an instance of a kernel, so in that case the vector is a kernel vector, as in the KPCA model.

Using a kernel has several advantages. Surface features do not have to be stored explicitly, thus we do not have to handle huge feature vectors. The kernel within the method can be replaced easily, so the method is modular. In fact, everything linguistic is in the kernel. The measure of similarity is entirely up to the researcher, as long as it fulfills the requirements of a kernel.

Continuous features can be simply read off the linear combination. The particularities of discrete features are discussed next.

2.4.4 Interpretation of points in the feature space: Pre-images

A point p in feature space is either given explicitly in surface features, or implicitly in terms of the principal components. The latter is equivalent to a representation as a linear combination of the training data, by multiplying the coordinates of a data item with α. Now we want to map p back into the original space, to its *pre-image*. This is immediate if that point represents a known data item. In other cases, an item has to be computed (Schölkopf and Smola 2002, ch. 18.1).

[17]Technically, the features in a kernel can be weighted.

One has to distinguish the case of continuous features from the case of discrete features. For every feature, p has a value according to the coefficients of the linear combination and the (known) values for this feature of each training item. If the feature is continuous (e.g., it represents a duration, or a formant value), the image in the original space can just take this value.

If however the original space only allows discrete values for that feature — as is the case with class membership — one has to pick a value. If they are mutually exclusive, their proportions will add up to 1. This can be seen as follows: it is true of the data mean, because it is true of every single item. Furthermore, there is no variation with respect to this, so the principal components only change the proportions of the different, mutually exclusive labels, not their sum. For an example, see 3.1.

The closer the test item is to instances of a certain class, the more its representation will rely on these, and thus it will receive more of that class label than of the other(s). Then a majority vote suffices: Take the label with the highest proportion. See the definitions in appendix D.

In linguistics, as in many other fields, one often deals with data more complex than just bundles of independent features. Explicit feature representations are unfeasible because of their length. Take strings as an example, and consider the string kernel defined in 2.2.3.1. The features of a sequence are its subsequences. These n-grams overlap (for $n > 1$) and therefore they cannot be evaluated independently. For example, the features `ab` and `ac` cannot be combined into a single word without introducing further bigrams.

Mapping a data item into the data space, which is a subspace of the feature space, means that the item's image is not necessarily lossless: Its actual feature representation is projected into the subspace. Conversely, a pre-image cannot be a perfect representation of a point in feature space, although for a different reason: The feature space is a continuous interpolation of admissible feature combinations, yet the actual data items need discrete feature values, at least for the types of features considered here (n-grams etc.).

A point in the feature space can assume any value of a feature. Consider a binary feature, that is, a feature which in any given data item is either present or absent, with no option in between. Both including it in and excluding it from the pre-image comes with a penalty. Say the feature's value is 0.6. Then including it adds 0.4 more of that feature than there should have been. Excluding it might even be the better option if that feature implies (in the original space) other features with even lower values. As an example consider a large n-gram, which implies all its substrings. Pre-imaging (the process of finding a pre-image) is always a compromise.

The infinitely many, yet enumerable items in a discrete data space will serve as a pre-image for a given point p to various degrees of goodness. Next, I will define this goodness based on the kernel, along with a general algorithm to search the data space iteratively for the one data item that fits best as a pre-image.

2.4.4.1 A general purpose pre-imager for discrete, complex data

This section describes a general purpose genetic algorithm for finding an optimal pre-image in a discrete space of structurally complex data. A more mathematical description can be found in appendix D.2.

Let the pre-image for a point p in feature space (expressed as a linear combination of the data set: $p := \overrightarrow{a}\zeta$) be defined as the item x in the data space such that x is closest to p. The distance in the feature space is calculated via the kernel as $\sqrt{k(x,x) + k(p,p) - 2k(x,p)}$ (see lemma 1 in appendix A). Thus its features do not have to be made explicit. If the distance is 0, then x and y are equal (by definition of distance).

Candidates pre-images x can be sorted according to their distance to p. One now needs a method to enumerate all candidates, i.e. all possible data items, to find the optimal one. Usually the data space will be infinite, so one also needs an abortion criterion which stops the search once a sufficiently close fit is found. However this strategy is very inefficient: enumerating the candidates consumes a lot of time. A dynamic search of the data space is better: the order of the enumeration is changed based on the evaluation of previous candidates. Basically, this calls for a genetic algorithm.

A genetic algorithm consists of two phases: mutation and selection. Mutation is the generation of new candidates based on previous candidates, and selection is the evaluation of candidates (via the distance). For simplicity, consider selecting only the one best candidate of each generation, mutations of which will form the next generation. Mutations are copies of the original with minimal changes. These minimal changes will be data-specific. For strings, but basically also for any structured data type, these might be insertions, deletions and substitutions (see the special-purpose string pre-imager in 6.2.1). They should be defined such that a candidate does not have too many descendants, and that the descendants cover all the relevant directions in which the pre-image could be improved: The immediate search space should be *dense*.

A strong assumption of this strategy is *monotonicity*. Only the best candidate is used to generate new candidates, and the search will be aborted if a certain new generation cannot provide any improvement over the previous candidate, declaring it the optimal pre-image. The minimal changes need to be general enough to avoid local maxima, although that risk always exists. Local maxima can be avoided by allowing several changes simultaneously (thus providing a look-ahead), at the cost of many more candidates to be considered. Monotonicity is a severe restriction, yet it provides an easy abortion criterion. In the applications of the string pre-imager (chapter 6) it proved reasonable.

Another important factor for the length of the search is the choice of the initial candidate. Most structural data types will provide a (near-)empty or minimal item. Starting there means building the pre-image from scratch. Depending on the task, a good guess of the initial candidate will massively reduce the number of generations necessary to find a winner, while at the same time avoiding all the errors (local maxima) that could be introduced by the longer search path.[18]

For lack of a perfect all-purpose pre-imaging method, one needs to find appropriate minimal edits

[18] As an example from biological evolution, take the appendix: Introduced at a much earlier generation, its removal would make us a better pre-image of the 'ideal' human. Yet the immediate search space (the next generation) is not dense: This minimal change is not included.

and initial items for every given application of the method. The markup pre-imager (5.2) is highly adapted to its purpose: A clever (actually, an obvious) choice of the initial item, and a minimal set of changes.

Asymptotic improvements ad infinitum are conceivable, but not very likely. The kernels used in this thesis, and most linguistic kernels in general, are *convolution kernels* (Haussler 1999): They compute similarity as a sum of the similarity of sub-structures. At some level, similarity is not recursively defined anymore, so it bottoms out to identity of atomic features. Asymptotic pre-imaging would mean adding more and more sub-structures, leading to ever larger candidates. Normalization of the kernel penalizes that. So the pre-imaging algorithm will usually terminate, although there might be pathological cases (choices of kernel function) in which this is not always the case.

2.4.5 Linguistic interpretation of factors

The space spanned by the data can be interpreted in comparison to the full feature space (as the set of 'legal' or 'grammatical' feature combinations; cf. also 7.2). The methods discussed so far (LSA, ICA, PCA) also give an orthogonal basis for this space, so the question arises how to interpret the basis vectors. In LSA, they are called the 'latent semantic dimensions', and they are intended to denote semantic concepts. ICA hopes to find linguistically meaningful semantic, morpho-syntactic and phonological features, depending on the surface features chosen. PCA extracts dimensions sorted by the amount of data variation they capture.

For non-kernelized PCA, I have discussed the shortcomings of principal components as linguistically meaningful entities in section 2.3.4.3. Other vectors in the data space might be more interpretable. This directly leads to rotations (2.4.5.1 here, and 4.2 later), as already exemplified by ICA, which is a rotation of an SVD. The additional problem with KPCA, both regarding the interpretation and rotations, is that the basis vectors are given as linear combinations of data items.

Dimension reduction of non-kernelized PCA (2.3.4.2) can be directly applied to Kernel PCA. There are no differences.

2.4.5.1 Factor rotations

Maximal variance is not the only criterion for determining a basis of feature space. The basis vectors can be rotated, retaining orthogonality, in order to meet other needs. As mentioned in 2.3.5 and 2.3.4.3, dimensions of maximal variance are not naturally interpretable. Rotations are meant to improve the interpretability; however the interpretation of independent components (ICA) is still difficult.

To my knowledge, all rotations work on explicit feature vectors. Therefore they are not directly applicable to kernel methods. In a later section (4.2) I will discuss rotations in more detail, address the problem of rotations in combination with kernel methods, and introduce a special rotation with respect to hidden features. This rotation maximizes the variance of the hidden features, and thus identifies dimensions which span a subspace of the data space, such that all variation of the hidden

features falls into that subspace. Therefore this rotation enforces a certain (pre-defined) interpretation.

One such application will identify two dimensions in surface variation of German nouns related to gender (4.4). They highly depend on the training set and are such that they do what they are made for; they do not mean anything beyond that. Thus not even this rotation leads to linguistically interpretable dimensions. In 4.5 I conclude that with a lot of effort one can enforce linguistic meaning (albeit only when it is pre-defined).

2.4.6 Comparison to Naive discriminative learning

Naive discriminative learning (Baayen et al. 2011) is similar to (W)KPCA. It also infers hidden features from surface features, and derives a transformation matrix to map data items to their hidden features. The main difference is that it does not use a kernel (and it is unclear to me how it could be modified in that way). The method is somewhat simpler, yet less flexible. For as long as explicit feature vectors are practical (explicit trigrams possibly count into tens of thousands), its results are very comparable to the KPCA method.

Next I show a special case of the model, which turns out to be identical to a treatment of the same data with the PCA method. This continues the example given in section 2.1.5.2. Table 2.35 shows the reconstructed hidden features when all words have the same frequency. The main difference to table 2.20 is that `lands` receives a full plural value (marked in bold). That value had been less important before, when `lands` had very low frequency compared to the other data items.

(2.35) Reconstructed hidden features, without frequencies:

	and	lass	sad	as	land	plural	lad	hand
hand	0.0909	-0.0000	-0.0909	-0.0000	-0.0909	0.0909	0.0909	1.0000
hands	-0.0909	-0.0000	0.0909	-0.0000	0.0909	0.9091	-0.0909	1.0000
land	0.2727	0.0909	-0.1818	-0.0909	0.7273	0.1818	0.1818	0.0000
lands	0.0909	0.0909	-0.0000	-0.0909	0.9091	**1.0000**	0.0000	0.0000
and	0.6364	-0.1818	0.1818	0.1818	0.3636	-0.1818	-0.1818	0.0000
sad	0.1818	-0.1818	0.6364	0.1818	-0.1818	0.3636	0.3636	0.0000
as	0.1818	0.3636	0.1818	0.6364	-0.1818	-0.1818	-0.1818	0.0000
lad	-0.0000	0.0909	0.0909	-0.0909	0.0000	-0.0909	0.9091	0.0000
lads	-0.1818	0.0909	0.2727	-0.0909	0.1818	0.7273	0.7273	0.0000
lass	-0.1818	0.6364	-0.1818	0.3636	0.1818	0.1818	0.1818	0.0000

For comparison, I implemented the same data set with a unigram[19] kernel in a KPCA model. The results are exactly the same (2.35). Given enough features,[20] both models can reliably recall the hidden features of the training data.

[19]Presence of features, not counts. This makes a difference for `lass` only. I implemented this by simply writing `las` instead of `lass`.

[20]Enough features means a invertible cue co-occurrence matrix, or a full rank surface feature (or kernel) matrix, in which case both models are exact, see appendix F.

In the general case with frequencies, I compared the results of Naive discriminative learning (section 2.1.5.2) to a Weighted KPCA model (2.4.2), using the same vector of frequencies. Everything else is the same as in the KPCA model. WKPCA's results are exactly those shown for Naive discriminative learning (2.20). Thus at least empirically in this example, the two methods are equivalent.

Naive discriminative learning is characterized by the feature co-occurrence matrix AA^T; KPCA operates on the kernel matrix A^TA. This is at the same time the main similarity and the main difference of the methods. As for the differences, the former is easier to calculate for few surface features and the latter for many.

Baayen et al.'s model is the latest word in explicit-feature models and should be preferred over e.g. networks. Both models do not need training; they compute the hidden features. There is the additional flexibility of kernels, both on the input and the output side. For models of inflection (6.4) explicit features (on the 'hidden' side) are not useful. Yet in many cases the question of explicit vs. implicit features remains one of personal choice, and of further research. The latter would uncover the close relationship of the models as witnessed by the results in 2.35.

2.4.7 Previous applications of KPCA

The only application of a kernelized factor analysis (a factor analysis the computation of which has been re-cast in terms of inner products) to a linguistic problem that I know of is Clark et al. (2006). The authors use a string kernel PCA to define the subspace all training items fall into, and consider all points within grammatical. The subspace is called a *planar language*. Clark et al. test this notion of grammaticality on a number of artificial languages ranging from regular to mildly context-sensitive complexity. Basically, if the implicit features have occurred before (in similar proportions), then a test item will fall into the hyperplane. Unwitnessed features and violations of feature proportions will lead to ungrammaticality.

This application is not even a classification, since it only knows one class: grammatical items. It learns from positive data only, and thus stands in contrast to Collins and Duffy's method, which has an explicit class of negative examples.

Costa Florêncio (2007) extends the notion of planar languages to tree kernels, but does not apply them. Further discussion of Clark et al. (2006) is found in section 7.2, where I apply tree kernels to tree representations of the languages studied in Clark et al. (2006).

2.4.8 Conclusion

The method outlined in this section sums up as follows. Linguistic objects can be arranged in a high-dimensional space according to their pairwise similarities, using KPCA. Similarity is calculated via a kernel function, which implements the researcher's notion of similarity for the given type of linguistic object. The kernel is defined on the surface features only, leaving the other features as hidden. Via its kernel vector (with respect to the training data), a test item can be projected into the

feature space, where it is expressed as a linear combination of training items. Because the kernel vector captures similarities, this embodies a model of analogy.

Assuming that the unaccessible, hidden features (class membership, grammatical features, and even its inflectional forms) behave analogously to surface features, they can be simply read off the linear combination. This is the main idea of this thesis: To infer those properties via *analogy*. If the features are discrete, and not continuous, it is necessary to find the closest possible value for the feature in order to obtain a valid pre-image.

I argue that this formalization of analogical reasoning is appropriate for linguistic models. It is data-oriented in that it abstracts from exemplar models. The method is modular, the calculation is exact, and the models are linear. The close connection between kernel methods and exemplar models was worked out by Ashby and Alfonso-Reese (1995) and is further backed up by Jäkel et al. (2008). In that sense, kernel-based models instantiate connectionism.

Although the model is used to classify data items, it does not directly define decision boundaries,[21] and thereby it differs from SVMs. The definiteness with which an item is assigned to a certain class depends on how close it is to instances of that class, and not on how far it is away from a boundary.

Still, this model of analogical proportion is opaque. It does extract the relevant information (as witnessed by its ability to perfectly reconstruct the training data), but it remains linguistically meaningless how it does so.

[21]The boundary would be given by the set of points in feature space with a loading of 0.5 for each class feature. In a multi-class scenario, this would be the set of all points the two highest class loadings of which are equal. These form a boundary, because in that case there is no unique winner.

Chapter 3

Classification

Classification is a well-studied task, and the prototypical application of machine learning. In 1.2.1, I have already introduced the concept: Given a training set and a test item, a model is to determine into which of two (or more) pre-defined classes the test item belongs. Being able to do so above chance level is taken as evidence that the model has learned something from the data. It is then probed for its internal representations, in order to interpret its modeling of the classes. In most cases, this means a comparison to rule-based or descriptive approaches.

As the prototypical application of machine learning techniques, classification is the first test case for the KPCA method (2.4.3). As a linguistic example, I chose nominal gender in German. Second language learners of German tend to ask for 'the rules' of gender assignment. To some degree, it is predictable from the surface form of nouns, as evidenced by native speakers' above chance agreement on novel words, but there are few observations which could be called 'rules' (3.3.1). Prediction cannot be perfect, since gender is an integral part of a noun; it distinguishes otherwise identical words and therefore it has to be learned just as any other property (e.g. a phonemic difference). Those properties are not fully independent, and it is their dependencies which are to be addressed.

I exemplify the KPCA method for classification in an extensive walkthrough (3.1). Afterwards I apply the very same method on a larger scale (3.2). It is necessary to keep the two apart, because a small model is no model of a native speaker's knowledge, and a large model cannot be walked through.

3.1 Walkthrough

It is the bane of data-driven methods that only a lot of data points will give reasonable results. So either a model performs badly, or it is unsuitable as a detailed example. This section demonstrates the inner workings of the method by a small example, not regarding performance. The following section (3.2.2) then treats the method as a black box, and applies it to a larger data set. The task and data format are identical; the two experiments only differ in size.

3.1.1 The data

As data, I choose 10 German nouns, which have some similarity and yet cover all three genders. I split them into 9 training items and 1 test item.

(3.1) Training data $\overrightarrow{a}$, all-lowercase and edge-marked, with gender annotation (m for masculine, f for feminine, n for neuter):

orthography	transcription	gender
#blau#	blaʊ	n
#traum#	traʊm	m
#tau#	taʊ	n
#baum#	baʊm	m
#raum#	raʊm	m
#sau#	saʊ	f
#frau#	fraʊ	f
#pfau#	pfaʊ	m
#stau#	staʊ	m
#saum#	saʊm	m

The mapping from words to hidden features can be written as a matrix (3.2):

(3.2) Hidden feature matrix B:

	blau	traum	tau	baum	raum	sau	frau	pfau	stau
masculine	0	1	0	1	1	0	0	1	1
feminine	0	0	0	0	0	1	1	0	0
neuter	1	0	1	0	0	0	0	0	0

The data set does not contain ambiguous words. One might add the masculine word “Tau” (*dew*), in order to have some ambiguity. However this would not add anything substantial to the example.

3.1.2 Procedure

The method divides into a training and a testing phase. In the training phase, a model of the training data is calculated. In the testing phase, it is used for classification. The training phase tries to find a new orthogonal basis for the space the data items lie in. This space is defined by their mutual distances/similarities, and the similarity is defined by the kernel.

Kernel and kernel matrix

I choose a bigram kernel (see 2.2.3.1 and appendix A.2). The kernel matrix (3.3) is the table of kernel values for all item pairs. (3.4) then is the centered kernel matrix: The kernel values for each item are 0 on average. See appendix B for the details.

(3.3) Kernel matrix K:

	blau	traum	tau	baum	raum	sau	frau	pfau	stau
blau	13	7	8	9	7	8	8	8	8
traum	7	15	9	10	12	7	9	7	8
tau	8	9	11	7	7	8	8	8	10
baum	9	10	7	13	10	7	7	7	7
raum	7	12	7	10	13	7	9	7	7
sau	8	7	8	7	7	11	8	8	10
frau	8	9	8	7	9	8	13	9	8
pfau	8	7	8	7	7	8	9	13	8
stau	8	8	10	7	7	10	8	8	13

(3.4) Centered kernel matrix:

	blau	traum	tau	baum	raum	sau	frau	pfau	stau
blau	4.740	-2.148	-0.259	0.629	-1.592	-0.037	-0.592	-0.148	-0.592
traum	-2.148	4.963	-0.148	0.740	2.518	-1.925	-0.481	-2.037	-1.481
tau	-0.259	-0.148	2.740	-1.370	-1.592	-0.037	-0.592	-0.148	1.407
baum	0.629	0.740	-1.370	4.518	1.296	-1.148	-1.703	-1.259	-1.703
raum	-1.592	2.518	-1.592	1.296	4.074	-1.370	0.074	-1.481	-1.925
sau	-0.037	-1.925	-0.037	-1.148	-1.370	3.185	-0.370	0.074	1.629
frau	-0.592	-0.481	-0.592	-1.703	0.074	-0.370	4.074	0.518	-0.925
pfau	-0.148	-2.037	-0.148	-1.259	-1.481	0.074	0.518	4.963	-0.481
stau	-0.592	-1.481	1.407	-1.703	-1.925	1.629	-0.925	-0.481	4.074

Principal components

PCA now calculates the principal components of the data set represented as linear combinations α of the training data, shown in matrix 3.5.

(3.5) Principal components α

	pc1	pc2	pc3	pc4	pc5	pc6	pc7	pc8
blau	0.054	0.249	-0.019	0.194	0.165	-0.264	0.118	0.120
traum	-0.150	-0.119	-0.047	-0.030	0.171	-0.211	-0.280	0.429
tau	0.053	-0.084	-0.096	-0.034	0.290	0.128	-0.079	-0.612
baum	-0.098	0.192	-0.072	-0.126	-0.102	0.427	-0.154	0.014
raum	-0.142	-0.033	0.037	0.017	-0.159	-0.196	0.464	-0.385
sau	0.087	-0.025	-0.069	0.029	-0.308	-0.248	-0.436	-0.168
frau	0.010	-0.085	0.218	0.293	-0.022	0.297	-0.087	0.077
pfau	0.084	0.016	0.226	-0.330	0.037	-0.086	0.056	0.108
stau	0.101	-0.109	-0.178	-0.011	-0.072	0.153	0.399	0.415
Eigenvalue	11.940	7.018	6.476	3.957	3.573	1.875	1.430	1.062

How to read them:

- Each of the 8 columns describes one principal component. Each row stands for one data item. The values (*coefficients*) denote how much of a data item is contained in a principal component.
- The principal components are orthogonal: The inner product of any two columns is always 0. This is a necessary condition for their independence.
- The principal components are sorted by the variance the data has in their respective directions, that is, by descending eigenvalue.
- The coefficients' squares do not add up to 1, so the principal components are not unit vectors in terms of data items, only in terms of features (see 3.6 and its discussion). Orthogonal unit vectors form an orthonormal basis of a subspace. Here, the subspace is the space spanned by the training data.
- Any rotation of this set of vectors is also an orthonormal basis of this subspace. This particular basis is special in that the first basis vector captures the maximal variance within the data; the second captures the maximum of the remaining variance; and so on.

These components are not aligned to exemplary items (prototypes). In the lower-ranked principal components there are some higher coefficients. Those can be interpreted as catering to the particularities of those few data items. The higher-ranked principal components identify more general distinctions within the data. Such general properties are not exemplified by single data items, but by a combination of many, which in turn will all have rather low coefficients, but add up to exhibit a certain property. Which these properties are can better be seen when features are made explicit, as shown for the first (3.6) and second (3.7) principal component. These representations are obtained by weighting the explicit feature listings of the data items (A) with the coefficients of the respective component.

(3.6) Features of the first principal component ($A\alpha_1^T$):

$-0.392\times$ `m, m#, um`
$-0.282\times$ `r, ra`
$-0.151\times$ `tr`
$-0.142\times$ `#r`
$0.102\times$ `st`
$0.155\times$ `ta`
$0.189\times$ `s, #s`
$0.392\times$ `u#`

(3.7) Features of the second principal component:

$-0.314\times$ `t`

$-0.239\times$ `ra, r`
$-0.205\times$ `#t`
$-0.195\times$ `ta`
$-0.135\times$ `s, #s`
$-0.120\times$ `tr`
$-0.101\times$ `st`
$0.193\times$ `ba`
$0.250\times$ `l, la, bl`
$0.442\times$ `b, #b`

How to read the explicit feature representation of principal components:

- Features are ordered by coefficients. Coefficients tell how much of a certain feature the principal component contains. Having a value of x on that dimension means taking x times the coefficient for all the features.
- I left out features with small coefficients (cut-off at ± 0.1), and collapsed features with identical coefficients to one line.
- The first principal component is a distinction between words ending in `aum#` and those ending in `au#`. However, this choice is not independent of other features. No linear combination of the training data can simply denote this one distinction. Other features come into play, and the overall variance is maximized. Here, this leads to the `aum#`-side also being loaded for features that occurred together with it: `r` and `ra` appear in two out of three words in `aum#`. Likewise, the other side contains features correlated with `au#`.
- Note that in the linear combination representation of the component (3.5), the three words in `aum` have negative coefficients, and no other. This directly leads to a cumulated high negative value for `m, um, um#` in the feature representation.
- `a, u, au` do not appear because they are never subject to variation (see 3.1.2).
- The second principal component is a further distinction of the data set under the premise that it be orthogonal to the first. It separates out the words with `b, #b` and `l`.
- The length of the component (sum of the squares of all coefficients, including the small ones I left out) is 1. Hence, the basis vectors are scaled to unit size.
- The vectors are orthogonal because the inner product of any two of them (sum of the products of all coefficients) is 0.

Also under explicit feature representations, the components are hard to interpret. This is due to the criterion of maximal variation. Alternatives (other orthonormal bases, obtained by rotation) are presented in chapter 4, section 4.2. Another way to visualize the components is to look at the values each data item takes on those dimensions. The first two are the most important, and so the data items can be plotted on a plane (figure 3.1).

How to read the visual representation:

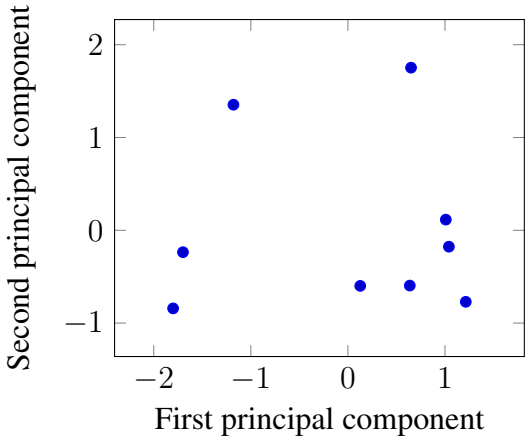


Figure 3.1: Feature space (first two dimensions) of the dummy data set

- The items to the left are the ones ending in `aum#`, as identified by the first principal component. Those two also fitting to the secondary features of the component enjoy more of it (and are further to the left). The negative value on the axis multiplied by the feature coefficients gives a positive occurrence of that feature, just as needed.
- The upper items are the ones starting with `#b` (second principal component). `#blau#` also has the secondary feature `l`, and thus is higher in the plot.
- All other items form a cluster, since they are only marginally distinguishable on the first two dimensions. The lower-ranked dimensions of variation will keep them apart, and account for their features and differences.

It seems quite arbitrary which feature combinations are important — and thus identified by higher principal components — and which are not. Here, this is merely an artifact of the data due to PCA being sensitive to particularities of the data set.

Center

Besides the principal components, PCA relies on the average of the data, the center. It is like a principal component in that it is expressed as a weighted sum of data items. The weights are all $1/9$, because this is how the average is calculated. It differs from the principal components in that it is not a dimension along that items can take different values. Its representation in explicit features is given in 3.8.

(3.8) Explicit features of the center:

$0.111\times$ `la, #f, bl, fa, ba, fr, sa, pf, #p, #r, st, tr, l, p`
$0.222\times$ `#b, ta, #s, #t, f, b, s`
$0.333\times$ `m#, ra, m, um, t, r`

$0.667\times$ u#
$1.000\times$ au, a, u
$2.000\times$ #

How to read the center:

- I collapsed lines with the same value. So the first line contains all features which occur once in the data set ($1/9$), and so on.
- Descriptions of points in feature space start at the center, having the values given in 3.8. Deviations from the center in a direction given by any of the principal components will change that.
- The bigram au and its unigrams appear once on average. This does not immediately say they occur exactly once per word. The centered data however has no variation at all with respect to those features, and hence there is no principal component which is loaded for these features. So no principal component can change the number of occurrences of au.
- The double occurrence of the sentinel is similar, however not an artifact of the data, but of the method, since every word has two of them.

Recall and precision

Now that the model has been built and extensively discussed, it is time to put it to use. For each data item, the model maps its kernel vector to a linear combination of training items (see appendix, A). The model is such that it has perfect accuracy on the training data. It maps all seen data items to themselves (more precisely: a linear combination containing 1.0 of themselves and 0.0 of all other items). Inferring the hidden features is trivial: They stay as they are.

Mapping an unseen test item cannot be trivial, since it cannot be mapped to itself. It is mapped to a feature space representation, the best one possible under the model. Input to the mapping is the kernel vector (3.9), here for the test item #saum#, output is a linear combination of the training items (3.10).

(3.9) Kernel vector[1] for saum given the data set ($\overrightarrow{k}(\overrightarrow{a}$,#saum#$)^T$):

	blau	traum	tau	baum	raum	sau	frau	pfau	stau
saum	7	10	7	10	10	10	7	7	9

(3.10) Linear combination of training data approximating saum

	blau	traum	tau	baum	raum	sau	frau	pfau	stau
coefficient	-0.185	0.239	-0.294	0.334	0.202	0.865	-0.227	-0.006	0.071

[1]The kernel vector is by definition (definition 4 in appendix A) a column vector. In matrix multiplications, it becomes clear whether it is needed as a column or a row vector. The formulas in the appendix take care of that by transposing vectors when needed.

One has to keep in mind that — given the little data the model has — the linear combination is only an approximation of the surface properties of the word saum. Let the accuracy of the approximation be measured as the *normalized* kernel value between the approximation and the actual item (see appendix A.1.3), in order to yield a value between 0 and 1. Here, is is 0.965. However, none of the features of #saum# are new, as all unigrams and bigrams of it are in the training set. Only their relative proportions are new, because in the model, the property of beginning with sau does not combine freely with ending with aum. Here is how to read the linear combination:

- The coefficients add up to 1.
- #saum# is mainly composed of #sau#, as it contains 86% of this word. Note two things: First, it is not fully contained because of their different right edge. Second, although their pairwise kernel value is the highest in #saum#'s kernel vector (3.9), it does not need nearly as much of the other words with which it shares the same similarity of 10. This is again to say that the kernel vector is not a good representation of a word, but the linear combination achieved via KPCA is.
- Other positive contributions to the word are those providing the final m. Together, they add up to 77.7% of an m. In that respect, the representation can only approximate the word. The model simply does not allow for free variation with respect to whether a word with nucleus au ends in m# because a lot more evidence is needed for establishing the independence of these two features. In all principal components, these features still are correlated.
- Finally, the m#-words introduced irrelevant n-grams, which need to be counter-balanced by components with negative coefficients. So #tau# counters the t of #traum#, and so on. Again, the approximation cannot be precise. It is just the best the model allows for.

As with the principal components, the linear combination can be made explicit:

(3.11) Feature representation of the approximation of the test item #saum#

$-0.234\times$ f
$-0.228\times$ #f, fr
$-0.223\times$ ta
$-0.186\times$ l, bl, la
$-0.054\times$ #t
$0.017\times$ t
$0.071\times$ st
$0.149\times$ b, #b
$0.203\times$ #r
$0.215\times$ r, ra
$0.223\times$ u#
$0.240\times$ tr
$0.334\times$ ba
$0.777\times$ m, um, m#

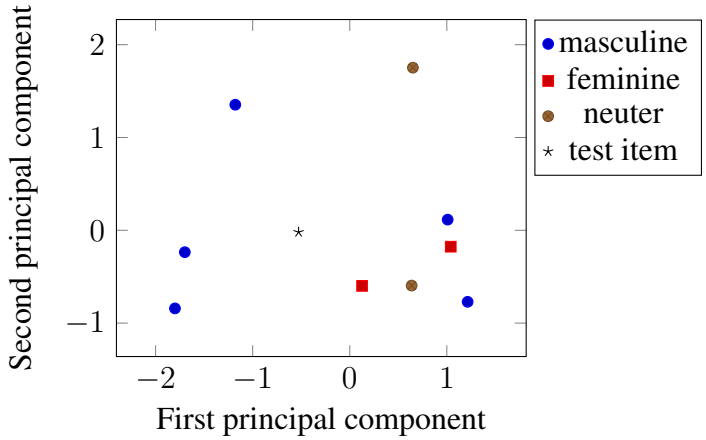


Figure 3.2: Feature space of the dummy data set plus test item

0.865× sa
0.937× s, #s
1.000× au, a, u
2.000× #

This is how to read the feature representation:

- Only a few things are certain about the word: its edges, and au. The rest is just the best approximation the model is capable of: It is hindered by the negative correlation of #s and aum# (and their respective sub-features) in the data.
- One can safely ignore features with negative coefficients, as the nature of the data does not allow that.
- Each positive coefficient can be taken as the amount to which the feature is contained in the (approximation of the) word.
- By taking a cut-off at 0.5 (where there luckily is a large gap), only and all those features present in #saum# remain.

A visual representation of the situation is given in figure 3.2. In addition to fig. 3.1, here the data points are marked for gender. The only notable correlation between the points and their gender is that all three words to the left (ending in aum#) are masculine. And so is the test word #saum#. However, it is drawn to the right by its close similarity to other words starting with #s. Accordingly, it receives mixed signals as to its gender. Each word contributes its gender according to its coefficient in the linear combination (the vector in 3.10). Summing over these contributions, one obtains the gender loadings in 3.12. This summation is defined as the multiplication of the items-by-hidden-features matrix B (3.2) by the (transposed) coefficient vector (3.10).

(3.12) Gender loadings:

$0.842\times$ masculine
$0.638\times$ feminine
$-0.480\times$ neuter

This is how to interpret the approximated gender value:

- First note that the values add up to 1, as every word has exactly 1 gender value. This is similar to every word containing the bigram `au` exactly once as seen earlier.
- Then wonder about the negative estimation for neuter gender, and the resulting high values elsewhere. This is a result from neuter words only being used with negative coefficients; an artifact of the data. Such output is not ruled out in general, but the extreme case witnessed in the example is clearly due to data sparseness.
- This prohibits readings where the gender loadings could be read as probabilities or as activations, which is tempting, since they add up to 1. Both readings do not accommodate negative values.
- Finally, it would be the prediction of the model that this word receive masculine gender, as contributed by the many masculine words in `m#`. Feminine gender is a close runner-up, but this is only due to a single data item very similar to the test item (`#sau#`).[2] Given more data, a newly seen word does not have to rely on single words to be represented. Then, and this is the general hope, the gender loadings of words beginning with `#sa` will level out, whereas the loading of words ending in `aum#` will always have a bias for masculine gender.

3.1.2.1 Discussion

In conclusion, there is a slight preference for the right gender of the test item. Of course, nothing is to be expected from such a small data set. The reader might want to consult his intuitions on another test item: `#flaum#`. In case he is fluent in German, he might consider a non-word similar to those in the training set. Again, the model would receive mixed signals from the onset and the of the coda of the word. Given enough data, patterns truly hinting at gender would become more stable, while other features of the word can be approximated neutral to gender. Given data exhausting the feature space (here: all combinations of onsets and codas), an independence of onset and gender could be established. However, the language experience of a native speaker hardly ever can prove such independence.

Connectionism assumes that nothing is independent, and that every language experience will shift the model, and so will the addition of new words here. It is a far shot to say that PCA derives a connectionist model, but I am claiming exactly that.

[2] Albright and Hayes (2003) argue against such influences. Only if a certain critical mass is reached, a small group of very similar words can spread their particular feature value to other words in their vicinity, in case the general vicinity is dominated by words with a different value. This is then called an *island of reliability*.

A drawback of the method is in approximating both surface and hidden features also with negative coefficients. A new word is never simply a *sum* of previous knowledge. On the contrary: The new word `#saum#` is presented along the lines of "just like `#sau#`, but without the final `u#`; very much like the group `#traum#`, `#raum#`, `#baum#`; but not at all like `#tau#`". To turn such a description back into an actual word requires a *pre-imager* (6.2.1). A more thorough interpretation of such representations is given at the end of the chapter (3.4).

Next, I will look at the large-scale performance of the model, and its ability to predict gender from surface forms.

3.2 Case study: gender in German

A case to study the prediction of abstract features on is nominal gender in German. In this section, I apply the method defined in 2.4.3 and exemplified in great detail in 3.1 to a large training set.

German gender is a good case to study. First, the co-domain of the mapping is a set of just a few, mutually exclusive class labels (`masculine`, `feminine` and `neuter`). Therefore it is a good instance of a classification problem.

Second, it has been a subject of study many times; Köpcke (1988, see 3.3.1) and MacWhinney et al. (1989, see 3.3.2) are explicitly discussed later. Citations therein consider gender assignment "entirely arbitrary" (Maratsos and Chalkley 1980) or "rule-governed, but exceedingly complex" (Mugdan 1977). The first position goes back to the fact that gender is an indispensable property of a German noun. It cannot be left underspecified, it sometimes is the only way to tell two words apart ("der See", *lake*, vs. "die See", *sea*), and except for dialectal variation every noun in the lexicon has exactly one correct gender.

New words may have any gender, however here if not rules then at least restrictions kick in. If a new word is the result of a morphological process (derivation, conversion e.g. of verbs), the final morpheme will mostly determine gender. In addition, there are pseudo-suffixes and other phonological (and also semantic) properties which prefer a certain gender. Cues of diverging predictive power lead to a model based on such cues (surface features), which I discuss in 3.3.1. Preferences are gradual, but altogether, gender assignment is not at all random. There is inter-annotator agreement far above chance level,[3] but the regularities of gender assignment are hard to capture, if at all. A more elaborate treatment of sub-regularities, exceptions, and regularities within exceptions will asymptotically improve the predictions of a rule-based model. It can be called "exceedingly complex", because there is no end to ever more fine-grained distinctions, down to single exceptional words. Exemplar models, or abstractions thereof (such as the KPCA model), are needed to capture such cases. Given the (partial) arbitrariness of gender, one needs to examine *how much* of the variation can be predicted using a formal model.

The data and its particularities are discussed in 3.2.1 before the actual experiment in 3.2.2. The

[3] This is my personal judgment both as a linguist and as a native speaker. However I was unable to find relevant linguistic studies supporting that claim.

model achieves up to 82% correct gender assignment, given enough data and informative enough features. Special attention is payed to morphological information (3.2.2.2).

3.2.1 The data

Theoretically, the training data is all the words a German native speaker knows, and the testing data would be loan words, neologisms and words simply unknown to the speaker. In the first two cases, native speakers need to agree on a gender, and they do so successfully, yet not perfectly by using their previous language experience. In the third case, even native speakers will err. Assigning gender is not a deterministic task.

In practice, I use data from CELEX (Baayen et al. 1996, see appendix G.2), divided into training and testing data, thus rendering a part of the words as unknown, while at the same time knowing their actual gender for automatic evaluation. Every noun is represented by a data item containing three pieces of information (3.13): its orthographic form, its phonemic transcription, and its gender. The actual representations the model uses are all-lower-case and contain the edge-of-word sentinels. For legibility, I will use the human-readable forms here.

(3.13)

	orth.	phon.	gender
human-readable	Gang	[ˈgaŋ]	masculine
machine-readable	`#gang#`	`#ˈgaN#`	1

There are 13873 non-compound nouns in the corpus. The gender distribution is as follows:

(3.14) Gender frequencies (types):

gender	all words	polymorphemic words
feminine	0.470	0.607
masculine	0.367	0.280
neuter	0.163	0.113

This suggests a simple baseline of 47% accuracy by always choosing feminine.[4] This high percentage is due to the many derivations in the database, many of which carry a morpheme determining feminine gender. This is backed up by the distributions for polymorphemic words: They are even more skewed towards feminine. This is however not an estimate of token frequencies.

3.2.1.1 Ambiguity

For several reasons, a word can have several gender values: dialectal variation or semantic distinctness. There is virtually no free variation. If there is a distinction relevant to the problem at hand, a distinction will be made, and if it is irrelevant, the problem will be ignored.

Gender sometimes distinguishes words, such as “der See, die See” (*lake* / *sea*). In the current model, this is bound to cause an error, but if the representation contained semantic information, the

[4] This is not a valid strategy in learning German, yet one that is commonly suspected to be employed by Italian native speakers.

words could be kept apart in recalling them, since the model is capable of perfect recall. Yet they are too close phonetically to induce a difference in prediction anywhere else.

Also, "die Leiter, der Leiter" (*ladder*, *leader*) differ in gender. Feminine words in -er are not uncommon, yet -er also serves to create masculine *nomina agentis* (cf. English: *ladder* is monomorphemic, *leader* contains a suffix -er). While semantic information would introduce a difference here, the real cause is in the morphological structure. Thus I add morphological information to the model as provided by CELEX.

(3.15)

orth.	phon	gender	morph.
Leiter	ˈlaɪ.tər	masculine	leit+er
Leiter	ˈlaɪ.tər	feminine	Leiter

"der Leiter" is morphological complex, containing the derivative morpheme -er bearing masculine gender and the verb stem "leit-" (to lead). "die Leiter" is monomorphemic. On this level of annotation, the two words are distinguishable. All that matters is that the kernel calculates a lesser similarity between the two than between one of them and itself.

Dialectal variation is exemplified by "die Butter" (*butter*; standard German) and "der Butter" (Swabian). CELEX only covers standard German, ruling out such a case. In appendix G.2, I list all words in the data having more than one gender (except where the two words are semantically distinct). This variation then is idiolectal, not dialectal: The collective native speakers of German have not agreed on a gender here. A given native speaker might have a clear opinion on a certain gender here and be surprised that others do not share it; or he might be unsure himself, mostly because he will never have used or even heard that word. To avoid confusion here I picked only the first gender given.

In another case CELEX misses the difference between "der Teil" and "das Teil" (*part/fraction* vs. *part/component*). Native speakers sometimes do confuse the two, although their meanings are in fact disjoint. Adding just a little noise to a model might provide exactly that, because of their phonological identity and semantic confusability.

3.2.1.2 Morphological annotation in CELEX

CELEX' morphological annotation also has its drawbacks. Even unproductive nominalizations without derivative morphemes are given as the corresponding verbal stem: The morphology of "Ausgang" (*exit*) is given as `ausgeh` (*to leave*). This stem lacks all phonological cues to gender present in the noun. Also, the prefix is not identified as a morpheme. The strive for verbal stems goes as far as positing `injiz+ion` for "Injektion" (*injection*), decomposing Latin derivations. The corresponding verb is "injizieren" (*to inject*; derived regularly from the Latin stem via the suffix -ieren), although this derivation is not transparent nowadays. English "to inject" derives from the past participle stem, and so does the nominalization "injection".

An especially curious case is found with the various nominalizations of "trinken" (*to drink*). "das Getränk", "der Trunk", "der Trank" (all meaning *drink*) and "die Tränke" (drinking trough) all have

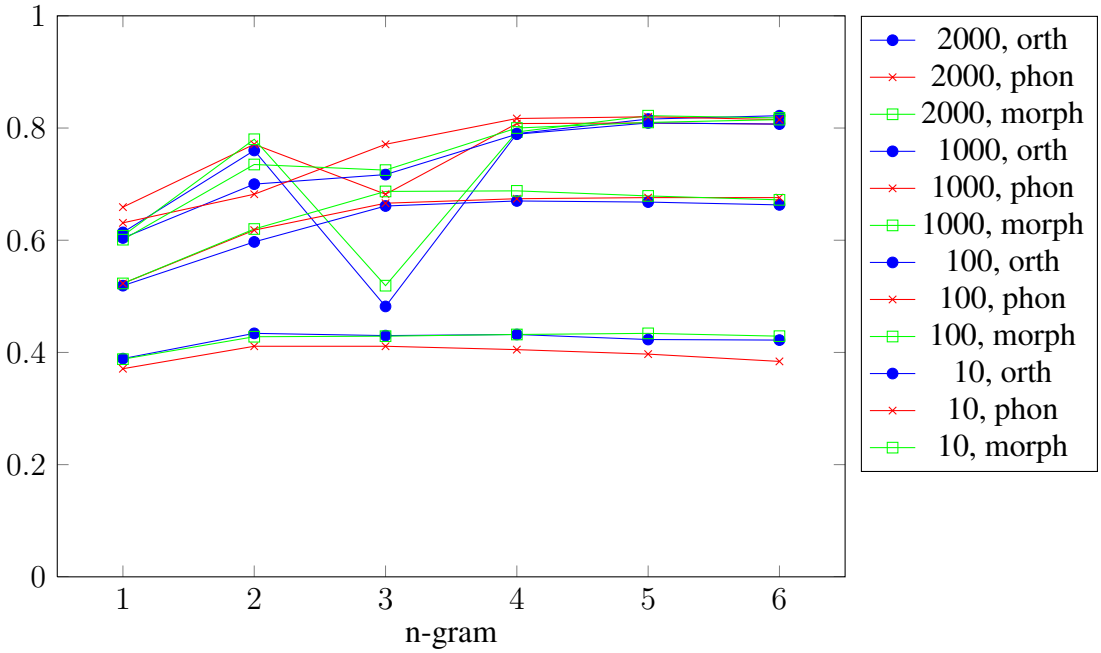


Figure 3.3: Gender assignment precision

`trink` (or `ge+trink`) as their morphological analysis. It is precisely the *type* of nominalization that determines gender. This is true for (almost) all derivational suffixes; vowel mutations yield monosyllabic words, which is a cue towards masculine (see the two different *ablaut* patterns, which nonetheless both end up being masculine; cf. 3.3.1).

The list could be continued with more design choices in CELEX which distort the prediction of gender. However I consider the deviations from the surface to be largely systematic, and I therefore expect that a model built on these representations will not be considerably worse.

Models are built on the orthographic, phonemic and morphological form. All are strings, so I will use a string kernel on both of them, more precisely, the n-gram kernel with varying values for n, as argued for in section 2.2.3.1.

3.2.2 Experiment

I conducted experiments with different training set sizes: 10, 100, 1000, 2000. 10 is supposed to yield some sort of baseline, because nothing will really be learned from 10 items. 2000 items are a bound set by the implementation, both in terms of memory usage ($>$ 1GB) and computation time. I looked at n-grams from 1 to 6. Remember that when testing a certain n, I also include all smaller n-grams as features. All combinations of training set sizes and n are run on 10 different, randomly chosen portions of the data.[5] The results given in 3.16-3.19 are average percentages of correct gender assignments on unseen words. Figure 3.3 shows the results graphically.

[5] 6 different values for the n-gram parameter, three different annotations levels, and ten-fold cross-validation lead to 180 separate KPCAs on 2000 data items each.

(3.16) Training items: 10

ngram	1	2	3	4	5	6
orth	0.389	0.434	0.430	0.432	0.423	0.422
phon	0.371	0.411	0.411	0.405	0.397	0.384
morph	0.388	0.428	0.429	0.432	0.434	0.429

(3.17) Training items: 100

ngram	1	2	3	4	5	6
orth	0.519	0.597	0.661	0.670	0.668	0.663
phon	0.523	0.618	0.666	0.674	0.676	0.676
morph	0.523	0.620	0.687	0.688	0.679	0.672

(3.18) Training items: 1000

ngram	1	2	3	4	5	6
orth	0.604	0.700	0.717	0.789	0.809	0.807
phon	0.631	0.682	0.771	0.817	0.820	0.815
morph	0.601	0.735	0.725	0.800	0.810	0.815

(3.19) Training items: 2000

ngram	1	2	3	4	5	6
orth	0.614	0.760	0.482	0.790	0.816	0.822
phon	0.659	0.771	0.682	0.808	0.809	0.807
morph	0.608	0.780	0.519	0.793	0.822	0.818

Interpretation and discussion of these results:

- The curves in figure 3.3 correspond, from bottom to top, to the training set sizes, from small to large. Thus performance increases with training data.
- The minimal model (10 items) reaches 40% accuracy.
- Even 100 training items improve the model considerably.
- There is only a small difference between 1000 and 2000[6] training items. The model reaches an upper bound here.
- Larger n-grams improve performance. The model stagnates after 5-grams, because larger features are so rare they are not useful in classifying unseen items.
- The small decline for higher values of n found with the small training sets is marginal, but might be attributed to overtraining: The model is not likely to see any of the larger n-grams from the training set again in the testing data. Such an effect would also appear for even larger n-grams with the larger data sets.

[6] And 3000, by manual inspection of single runs with a set this large.

- There is no apparent difference in the performance of models trained on orthographic, phonemic and morphological data, respectively.
- Most striking in the results is the drop in accuracy with trigrams at 2000 training items. It appears only with trigrams and large training sets. I have no explanation to offer, not even a highly speculative one.

A gender assignment is considered correct if the the gender value with the highest loading is the correct one. It does happen that this highest loading is no more than 0.4. Remember that the loadings add up to 1, since every noun has exactly one gender value. A loading might also be as high as 1.2. This is the case for points beyond the cluster centers. Other values then are below zero, in line with the sum of 1. Another way to look at the accuracy is to add the loadings assigned to the respective correct values. This undoes the effect of taking the gender with highest loading and discarding the others, by taking them at their original value. This also reintroduces the lost loadings of the correct gender in mis-classifications. With a 4-gram kernel and 2000 training items, the sum of loadings of the correct gender (of 100 training items) is 65.5. That means the assignment is still twice as good as chance (33%), and better than the baseline "all feminine".

The confusion matrix (3.20) shows how often which gender was classified as which; based on a model from 2000 training items, with a 4-gram kernel on the orthography, tested on 1000 items.

(3.20) Gender confusion matrix:

	classified as			
	masculine	feminine	neuter	accuracy
masculine	281	33	33	0.810
feminine	42	449	9	0.898
neuter	55	11	87	0.569

Feminine nouns are classified correctly most often. This is facilitated by the many morphologically marked feminine nouns. Neuter nouns are often (55) mistaken for being masculine. This is not unexpected, considering the intuitions of native speakers, also witnessed by the large list of words with masculine or neuter gender (see G.2): 62 words in the corpus are ambiguous between the two. Remember that this is not intra-, but inter-speaker variation. This is evidence of an inherent confusability of the two. Also, 33 masculine nouns are classified as neutral, adding up to 88 confusions. The 75 (33+42) cases of masculine/feminine confusion are more than one would expect by the same reasoning: Only 14 words are ambiguous between the two. Feminine/neuter confusion is lowest ($9 + 11 = 20$), just as expected. In this respect, the model behaves reasonably in its errors.

3.2.2.1 Interpretation of the principal components

The model constructs a *feature space*: All items are placed according to their pairwise similarities, as given by the kernel. Each test item is mapped into feature space, to a point where its pairwise similarities to the training items are best met. From this, I inferred its gender. However this point

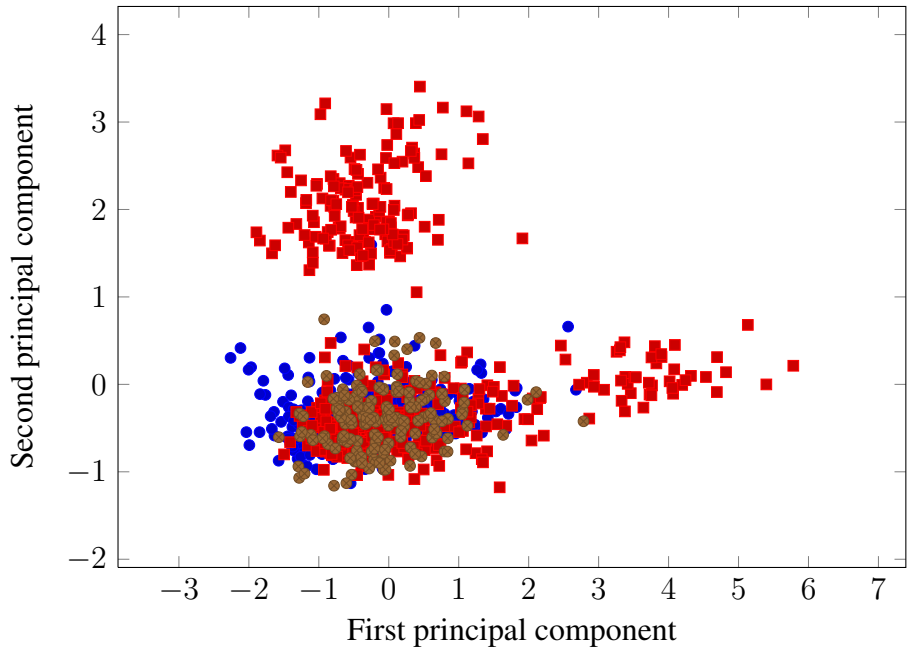


Figure 3.4: Plot of 1000 training items along the first two principal components

is not meant to be a representation of gender, only one of the surface form, in terms of the kernel's implicit features. This section looks at the principal components in more detail, but only to conclude that while at first glance they might contain relevant information, they in fact are linguistically void. Also refer to section 2.3.4.3.

First remember that principal components are sorted by descending variance, that is, the training items are dispersed more widely on the higher components. In the naive interpretation of PCA, these are the more important ones, whereas the lower ones only capture minuscule deviations of training items from the expected form, pinning them down to their exact values. I will show in section 4.1 that the lower components can be discarded without loss of much information. The best way to visualize the feature space then is to plot all training items along the two most important principal components, i.e., the first two, as is done in figure 3.4.

The plot basically shows three clusters: There is a main cluster centered around the origin, containing words of all three genders. Color represents gender: red for feminine, blue for masculine, and brown for neuter. Two clusters stand out, one along each of the two components. Notably, there are almost only feminine nouns in these clusters. As we will see, this is an artifact of the data. The following listings show the highest absolute feature loadings of the two principal components.

(3.21) First principal component:

$-0.2247\times$ `@`
$-0.1927\times$ `r`
$-0.1364\times$ `@r`
$-0.0974\times$ `r#`

$-0.0947\times$ @r#
...
$0.1077\times$ atsi:
$0.1130\times$ ats
$0.1329\times$ tsi:o:
$0.1352\times$ si:o:
$0.1352\times$ si:o:n
$0.1367\times$ i:o:n#
$0.1370\times$ o:n#
$0.1406\times$ at
$0.1421\times$ i:o:n
$0.1422\times$ i:o:
$0.1435\times$ n#
$0.1446\times$ o:n
$0.1563\times$ tsi:
$0.1611\times$ si:
$0.1854\times$ o:
$0.2022\times$ ts
$0.2322\times$ a
$0.2621\times$ i:
$0.2669\times$ n
$0.2820\times$ s
$0.3873\times$ t

(3.22) Second principal component:

$-0.2190\times$ @
$-0.1032\times$ @#
...
$0.1310\times$ rUN
$0.1310\times$ rUN#
$0.1379\times$ rU
$0.1456\times$ i:
$0.1601\times$ r
$0.3529\times$ UN#
$0.3546\times$ UN
$0.3571\times$ N#
$0.3675\times$ N
$0.3791\times$ U

The first component can be summarized as a distinction between words ending in @r# and those ending in tion# (with a strong preference for ation#). Both happen to be gender determining derivational morphemes. However, this interpretation is distorted: Words do not decide whether they have this *hidden feature* (interpreting the principal component as such), they take very different values along the dimension, admittedly with a large gap. If the dimension denoted the morpheme -tion, then all words with this morpheme should have the same value along the dimension. This is clearly not the case.

The centers of mass of the clusters are around the values -0.5 and 2 on the first dimension. Again, there is no interpretation of having the feature (1) or not (0). This is impossible due to centering: The origin of all dimensions is the mean of the data. This pretty much prohibits any interpretation in which a component denotes a feature which is either present (once or several times) or absent.

There is a simple reason for the shape of the first component: Given the many derivations in `tion#`, it happens to be most informative of all to know to which extent a word has these features. `@r#` then is sort of the opposite: the diametrically opposed pole to `tion#`. Were frequencies different in the corpus, other distinctions might have been more informative than this one. The component denotes neither of the two endings; it is merely a reflection of n-gram frequencies in the data.

A very similar picture is given by the second principal component. It contains n-grams related to `UN#` on the one hand (I have listed only a very few of them), out of which `rUN#` is particularly loaded, and `@#` on the other hand. The different loadings, the dispersion of data points along the dimension, the connection to to frequencies in the data, they all argue against a direct interpretation.

These two dimension are highly correlated with morphemes, because those morphemes are highly frequent. This is the only reason. From these morphemes follows the correlation with feminine gender. They do not denote feminine gender, for several reasons, most importantly for that there are already two dimensions which could claim that job. The correlation between components and gender is maximized in section 4.4 via rotation of the basis of feature space, but still there is no easy interpretation.

The dimensions of maximal variation within the set of German singular nouns are not hidden features in a linguistic sense (gender, morphemes), although they highly correlate with them due to biases in the data. Neither are they building blocks, because of the continuity of values along the dimensions: Building blocks are something discrete (cf. 2.3.4.3).

3.2.2.2 Morphological Information

In the previous experiment, there was no observable benefit from morphological analysis. The morphologically disambiguated cases (“Leiter”) are too rare. I repeated the experiment on morphologically complex nouns only. There are 6642 such nouns in the corpus, out of which I randomly chose 1000 for training and 100 for testing. A model was built using 4-gram kernels on orthography and morphology, respectively. Accuracy goes up to 94/95%. There is no advantage for the morphological annotation, because there are no words with competing morphological analyses in the data. One analysis of “Leiter” is monomorphemic, and such cases were not included in the data. As said, they are too rare anyways. 3.23 shows the remaining errors.

(3.23) Analysis of errors using a kernel on the morphology (6 out of 100):

word	morphology	gender	predicted	status
kastrat	kastr+at	m	n	ambiguous morpheme
streusel	streu+sel	m	n	ambiguous morpheme
handbreit	hand+breit	f	m	headless compound
disputant	disput+ant	m	n	error
zusprechung	zusprech+ung	m	f	annotation error
komponente	kompon+ente	f	m	error

Analysis of errors using a kernel on the orthography (5 out of 100):

word	morphology	gender	predicted	status
kastrat	kastr+at	m	n	ambiguous morpheme
getränk	ge+trink	n	f	error
streusel	streu+sel	m	n	ambiguous morpheme
zusprechung	zusprech+ung	m	f	annotation error
verhältnis	verhalt+nis	n	f	ambiguous morpheme

Several errors are accounted for by the gender ambiguity of the morpheme. Not all derivational morphemes uniquely specify gender. Another error is a clear mistake in the database. In `kompon+ente` (*component*), the feminine pseudo-suffix -e cannot outweigh the 4-gram `+ent`, which clearly calls for masculine gender. "Handbreit" (*a hand's breadth*) is a strange compound with an adjectival head. The gender seems to be inherited from the noun "Hand" (*hand*), something which the model was entirely unprepared for, and rightfully so. And finally, some errors remain without reasonable explanation. Curiously, it is not the morphological kernel which mis-classifies the aforementioned "Getränk".

Gender can be learned from morphologically complex nouns more easily because of gender-determining derivational morphemes. Having the morpheme boundaries then is of no advantage.

3.2.3 Weighting the data

The previous experiments have treated all input data alike, of equal weight towards the model. Next, I will show Weighted KPCA's (2.4.2 and appendix C) performance in comparison to plain unweighted KPCA. I applied "add 1"-smoothing to the data: Adding 1 to each frequency in order to avoid zeros. An item with zero frequency would otherwise result in a kernel matrix of reduced rank. The results (for a slightly smaller setting than above, in terms of parameter combinations) are given in 3.24-3.25. The unexplained problem with trigrams is reproduced.

(3.24) Training items: 1000

	ngram	2	3	4	5
orth	weighted	0.544	0.640	0.694	0.706
	unweighted	0.714	0.640	0.694	0.706
phon	weighted	0.580	0.694	0.736	0.726
	unweighted	0.646	0.694	0.736	0.726

(3.25) Training items: 2000

	ngram	2	3	4	5
orth	weighted	0.614	0.460	0.732	0.728
	unweighted	0.754	0.464	0.734	0.728
phon	weighted	0.660	0.670	0.790	0.784
	unweighted	0.764	0.668	0.790	0.784

WKPCA appears to be at a disadvantage for limited n-grams. Note that WKPCA is only equivalent to a KPCA over an expanded kernel matrix, so they may very well be different in outcome. Here, bigrams do not suffice to recognize relevant patterns, so in the weighted version the description of a test item will over-use more frequent words because those similarities weighted higher. Apparently with limited features it is better not to rely on frequencies, but to take all data items into account alike.

For fully informative n-grams, KPCA and WKPCA achieve the same performance. After all they both describe the same data space. They merely differ in the basis (the principal components) they derive. Using the full rank, that is, making use of all available information, the whole space is preserved and thus there can hardly be a difference. The anticipated advantage of WKPCA would have to be found in its first principal components being even more important than KPCA's. Therefore I compare KPCA and WKPCA again in section 4.1, under reduced dimension.

3.2.4 Conclusion

Using the KPCA method, gender of previously unseen German nouns can be predicted with over 80% accuracy given as little as 1000 randomly chosen nouns. A comparison to other methods on similar tasks is given in section 3.3. For a comparison with human performance on non-words I could not find any study; the one by Schwichtenberg and Schiller (2004) is too specific. They show in an empirical study, that semantic classes heavily influence human gender assignment to non-words, e.g. predators being mostly masculine. These effects can override phonological cues. That means gender correlates with semantics. This is also found by Konishi (1993). Köpcke and Zubin (1984) find that the gender differences in words ending in '-mut' (a cognate of *mood*) correlate with extroversion (masculine) / introversion (feminine): "der Hochmut" (*pride*) vs. "die Demut" (*humility*).

I did not take into account semantic information. A possible way of doing that is deriving similarity measures from an ontology, reformulated as a kernel and added to the surface kernel.[7] From this one would expect as model capable of similar behavior as reported in Schwichtenberg and Schiller (2004): Assigning different gender to surface-identical words for semantic reasons. As the model gives relative loadings for each gender, one might even hope to model the difference between clear and not so clear cases, as evidenced by human judgment.

The sample of performance values is too small and the models are too susceptible to the data, so it makes no sense to check their difference for statistical significance. Fixing the kernel and the

[7]The sum of two kernels is again a kernel, see appendix A.1.1.

data set size, performance still varies given different training and/or testing sets. In order to interpret this variation, the data would have to be balanced in various ways. This would not add anything of interest to this thesis. I chose to just give the average values of ten runs, ensuring comparability by using the same data selection with each kernel. Also, the absolute performance rates are not to be taken too exactly, since they depend on the data.

This model shows that analogical reasoning (as in an exemplar model) in assigning gender can be made mathematically explicit using kernel methods. It however does not imply a default. Such an interpretation would be part of the pre-imager: In case the model's answer is inconclusive (say, the highest loading for a gender is below 0.5), choose masculine.

While I do not provide further models as part of this thesis to back up the claim that this method for analogical classification can be applied generally, its generality and flexibility very much suggests so. The trick is in the appropriate characterization of similarity, which has to be encoded as a kernel. Semantic cues etc. can be added to the kernel as features, so by this flexibility of kernels the linguist can define similarity to his or her needs. And already with a simple, general-purpose n-gram kernel, the results are competitive.

3.3 Comparisons

This section methodologically compares the KPCA method for gender assignment to similar approaches from the literature.

3.3.1 Rule-based models

Linguistics as any science welcomes parsimonious models. If there is a task to map data points (words) onto abstract features, then the smallest possible model is that of a rule, thus identifying some abstract, underlying truth behind the data, like a natural law. Now the whole point of connectionism is that language simply does not work this way. Abstract classes as posited by linguists are meaningful, from which it follows that they are not redundant. Hence rules will never describe the full data set. Nevertheless they are a good approximation, because the can correctly and parsimoniously handle large, regular portions of the data.

Rules take the form of deriving an abstract feature from some observable variable. The larger the domain of a rule, the more likely it is to have exceptions. It is in the nature of linguistic data, that there will be exceptions, and regularities among the exceptions. Thus a rule can be considered a reasonable abstraction over the data even if it partially makes wrong predictions, at the benefit of being concise. Rules are thus violable or overridable by other rules.

Such a model for German gender is attempted by Köpcke (1988), who identifies the most reliable surface cues for gender assignment. These are already a step away from rules and towards exemplar-based models. The cues are (as cited by MacWhinney et al. 1989):

(3.26) Phonological cues:

masculine: umlaut, tr-/dr-, -əl, -n, CV-, CCV-, CCCV-, -VC, -VCC, -VCCC, sC-/ʃC-, monosyllabic

feminine: -[frikative]t (e.g., “Bucht”), -ə

masculine/neuter: -s

Here C stands for any consonant, V for any vowel, [fricative] for any fricative. Hyphens identify the cues as word-initial or word-final.

Observe that these are merely cues, not rules. Whereas in other classification problems, single rules can capture large portions of the data, this is hardly the case for gender of German nouns. Some cues are quite reliable, such as -ə, but “Riese” [riːzə] is masculine. For virtually any cue, there are numerous counterexamples. Enumerating them or even statistically evaluating the validity of each and every phonological cue would not significantly improve support for this conclusion: Cues are not rules.

The cues exert a default for masculine: All non-empty onsets and codas counts towards it. For many words, these are the only cues applicable. Masculine cues are also redundant, e.g. the CCV-pattern is also instantiated by the cues tr-/dr- and sC-/ʃC-. Other cues are contrary: -[frikative]t is a relatively good cue for feminine, but it is also an instance of -VCC (masculine).

Then, some words exhibit many cues: “Schrift” [ʃrɪft] fulfills four cues for masculine (ʃC-, CCV, -VCC, monosyllabic) and one for feminine (-[fricative]t). Since it is feminine one could conclude that a hierarchical ordering or a relative weighting of the cues is called for. Furthermore, it is unclear how to evaluate the only cue for neuter (-s), since it is also a cue for masculine. To avoid further discussion, I grant the description a best case application: Whenever I (manually) evaluate words according to the cues, I will pick just the ones pointing to the correct gender; leaving aside whether this could be formalized.

(3.27) Morphological cues: suffixes determining gender

masculine: -ling, -ent, -er, -eur, . . .

feminine: -ei, -ie, -ik, -in, -ion, -ität, -sis, -ung, . . .

neuter: -lein, -ment, -ell, -chen, -en (de-verbal nouns), -um, . . .

Unlike the phonological ones the morphological cues are given in their orthographic form. These lists are not exhaustive: When in manual evaluation I encountered other suffixes in the data, I added them accordingly. Not all of them are productive, and many of them derive from foreign suffixes (-eur from French, many others from Latin and Greek). Some of them resemble non-suffixes: There are regular words ending in -er (“Leiter”, f.), -ei (“Brei”, m.), -ung (“Dung”, m.) etc. where this letter/sound sequence is not a morpheme. Apart from that, morphemes are quite reliable. Some morpheme such as -är, -tum, -nis, -mut did not enter the list, since their gender is not deterministic (see also Köpcke and Zubin 1984).

Correctly applying the morphological cues thus requires a correct morphological analysis of the word, plus inherent knowledge about even non-productive pseudo-morphemes from French, Greek and Latin. Given that, they are the most reliable source of gender information.

Finally, there are a few semantic cues.

(3.28) Semantic cues:

- sex (masculine/feminine)
- young human or animal ("Kind"): neuter
- hyperonyms ("Tier"): neuter

These cues are less reliable than they are reasonable. Although gender originally reflected sex, it is nowadays a mostly abstract grammatical feature. The cues are also in conflict with each other: "Junge" (boy) is masculine, "Junge" (young animal of certain species: cub, hatchling) is neuter. Note the clash with the cue -ə for feminine. Anyone who has ever heard of German gender will know "Mädchen", and will be able to figure out the feature clash for himself.

Semantic cues hardly ever apply, since most things are genderless. I did not use semantic cues in manual evaluation.

Comparison

I compared the application of the cue set with the KPCA method, with 1000 training items and a 3-gram kernel. I evaluated both on the testing data in G.2.1. Evaluation of the cue-based method was done manually, since most cues are hard to formalize, their relative strength is undetermined, the morphological cue list was incomplete etc. In case of a clash or doubt I used the cues leading to the correct gender, thus granting the model the possibility of a specification without further complications. In particular, I made the following choices:

- All gender determining suffixes were used, even if they not appeared in the list.
- I assumed full morphological analysis, and knowledge of all unambiguous morphemes, even if they were not listed in the original cues.
- The cues are far from being mutually exclusive. If they differ, I relied on the "correct" ones.
- Semantic cues were not applicable within the testing data.
- The phonological cues exert a default for masculine, and thus I chose masculine if all else failed.

The last choice is in contrast to the most frequent gender (feminine). This is the true strength of the rule-based model: Many correct assignments were due to the default. Many feminine words are derivations, and these are already covered by the morphological cues.

On the test list (appendix G.2.1), Köpcke's cues, applied as described above, correctly classify 81%, whereas the KPCA model only gets 77%. 14 errors were shared, and neither model predicts a subset of the correct predictions of the other. The KPCA model sometimes has the correct gender as a close runner-up (but this might also be the case for false values in correct predictions). For example, "Orientale" (*Oriental*) has very similar values for masculine and feminine, and in fact it can be both. Such an interpretation could also be possible under certain weightings of the cues.

The rule-based model profits from guaranteed correct results on gender determining morphemes, however it has no way of dealing with the ambiguous ones. A data-oriented method first needs to see each morpheme at least once in the data. This requires either a large or an intentionally balanced data set.

The KPCA model can deal with incorrectly assigned words by including them into the model, just as a rule-based model can keep a list of exceptions. The advantage of a KPCA model here is that all items are treated alike, and that so-called exceptions are abstracted over in the very same way as all other items are. This theoretically allows capturing regularities on all levels.

Both methods fare badly with regard to neuter gender. Köpcke's phonological cues virtually never point to neuter, and neuter words mostly bear the cues for masculine. Both methods fail to predict an annotation error ("Haft" is indeed feminine, see the list).

Conclusion

Some cues (tr-/dr-) are very specific and exemplified only by a few words. They could only be learned given huge amounts of data containing at least a few positive instances of the pattern. Identifying subregularities at this level of detail is in fact already data-oriented. tr-/dr- has never been claimed to be a regular marker of masculine gender.

All phonological cues from Köpcke are violable, not absolute. It remains unclear to me how cue conflicts are resolved, be it with a weighting or a hierarchical ordering. The nature of the problem rather calls for a data-oriented approach: There are regularities and sub-regularities down to single exceptions, but no rule with a clear domain of application. This is true for the phonological and semantic cues, however most derivational morphemes do determine gender. Where this is not the case, the distribution of two genders for one morpheme was suggested to depend on subtle semantic similarities, dividing the words derived by the morpheme into two groups (Köpcke and Zubin 1984).

With its hand-selected cues, full access to morphemes, the strong default for masculine, and good will in the application of the cues (which cue takes precedence over which), the rule-based model has less errors than the KPCA model, both on monomorphemic and polymorphemic words. Only with neuter words, the latter has an advantage, due to the lack of non-morphological cues for neuter. In theory, data-oriented models are able to learn all the cues as abstractions over the data. So why does the proposed KPCA method fail to do so? First of all, with more data it approaches performance at around 80%, just as the cues. With even more data, covering the full experience of a native speaker, as also underlies the cues, this is not likely to rise significantly. Higher accuracy (or inter-annotator agreement for nonce words) could to my reckoning only be achieved by more levels

of annotation, e.g. explicit semantic information (see Schwichtenberg and Schiller 2004 for a related study). A kernel on this other level of annotation can be added to the surface kernel. Kernels enjoy modularity here.

If analogy by surface similarity is the right approach to gender assignment, then a n-gram kernel is still only a first shot at a measure of similarity. Fortunately, the KPCA method itself does not depend on the choice of kernel; other measures can be plugged in, modeled according to what by psycholinguistic insight constitutes similarity. Given its generality and flexibility, the KPCA model holds the promise of being able to do better than rules. As it stands, it is only almost as good, but mathematically precise and generalizable to all other classification problems, and open to improvement (with different kernels).

3.3.2 Neuronal Networks

Networks are a possibility to model the mapping of words to their hidden features (2.1.4). A study both using networks and treating a problem close to German gender assignment is MacWhinney et al. (1989). As input nodes, they compare Köpcke's cues to explicit surface feature vectors. Their output comprises number, case and gender. This is not rendered explicitly, but implicit as the task is to choose the correct definite article. The 24 possible combinations of 2 numbers, 4 cases and 3 genders are covered by only 6 different forms of the definite article, as seen in 3.29. As gender is not distinguished in plural, this immediately boils down to only 16 combinations.

(3.29) German definite article:

	masc.	fem.	neut.	plural
Nominative	der	die	das	die
Genitive	des	der	des	der
Dative	dem	der	dem	den
Accusative	den	die	das	die

Their training and testing data is derived from a 80,000 words corpus of spoken German, by taking all noun lemmas which appeared at least 15 times (102 different nouns) and placing them in different contexts. The contexts are represented by 17 input nodes encoding different word order configurations, prepositions and verb types. There are 20 possible contexts (that is, 20 combinations of the context features). Each word is shown in a number of randomly chosen, yet different contexts according to its relative frequency, ranging from 1 to 17 (out of 20). This yields 305 training items (noun-context combinations) in total.

I assume this training set is lexically far less diverse than the word list I extracted from CELEX (appendix, G.2), for two reasons. First, it is taken from spoken language, which would reduce the amount of technical terms etc. Second, the frequency cut-off reduces the set to the 102 most frequent nouns, all of which presumably are in the core lexicon. Online word lists[8] are usually

[8]E.g. provided by the University of Leipzig, `http://wortschatz.uni-leipzig.de/html/wliste.html`

derived from newspaper text, as evidenced by many words typical of news, which however rarely appear in spoken language.[9] For the learning task this means the data is rather homogeneous, quite contrary to CELEX.

Testing data are the unseen contexts for all words, and new words. Given 102 words times 20 contexts, minus the 305 training combinations, this leaves 1735 testing items. The new words are the next 48 by frequency, that is, those appearing 10 to 14 times in the corpus. They were put into all 20 contexts, yielding 960 additional training items.

Models 1 & 2

Their models 1 and 2 employ Köpcke's explicit phonological, morphological and semantic cues (see 3.3.1). A node for each is activated if the cue is present for a given word, and not activated if absent. An artificial means of full distinction is implemented by 11 nodes: Different combinations of activations encode each lemma.

The context is specified by further input nodes. The model has 20 hidden nodes to learn from the cues and lemma codes (every node is connected to all 20 hidden nodes), and 10 hidden nodes to learn a compressed representation of contexts (again, fully connected). The two groups of hidden nodes then are brought together by a second hidden layer of 7 nodes, which in turn are connected to the 6 output nodes, one for each of the forms of the definite article (*der, die, das, dem, den, des*).

The model is trained on the 305 training items, until it masters that set (for details, see there). It achieves 92% accuracy on the first training set (unseen combinations) and 61% on the second (unseen words). The first result suffers from words with unknown gender: Not all observations (context as input, definite article as output) identify gender. For example, a plural environment will never reveal the gender of a word. In the first test set there are all unseen contexts, hence also the ones requiring knowledge of gender. In these cases, even a model that performed perfectly on the training set is likely to fail.

The second training set tries to disentangle the contributions of two sets of input nodes: The lemma code and the cues. It might be that the model only learns from the lemmas, and does not need to look at the cues at all. The model's performance on unknown words, for which there is no known lemma code, drops to 61%. This suggests it learned little from the cues. In order to further test this, a second model was set up, forcing the model to learn from the cues by omitting the lemma codes. Performance significantly dropped even when accounting for the cases where lemma codes were only meant to disambiguate words of different gender, yet with identical cues. Thus MacWhinney et al. conclude that cues are not sufficient.

In a third model they take a step towards greater flexibility of input representations, thereby also alleviating the need for lemma codes, because words will become fully distinguished by their surface features.

[9] Out of the 20 most frequent (see `http://de.wikipedia.org/wiki/Liste_der_häufigsten_Wörter_der_deutschen_Sprache`), these would be: *Prozent, Million, Deutschland, Berlin, Unternehmen, Milliarde, Regierung, Land*, with several others being common in everyday language, yet still overrepresented in newspaper text.

Model 3

The third model employs vectors of explicit surface features as inputs. These replace the morphological and phonological cues; the semantic cues and of course the context representations remain. The model achieves to predict the correct article in 93,6% of first set of test cases (111 errors out of 1735 unseen testing words). No results are given for the second training set, the unseen words.

The words are input as a vector of features. Each feature is either present or absent, and if present activates its input node. The features chosen by MacWhinney et al. directly encode the phonological form of a word. The vector is a concatenation of 13 feature vectors, each standing for a phoneme in a certain position:

1. Up to three consonants in the initial consonant cluster,
2. up to two vowels in the first vowel cluster,
3. up to three consonants in the medial consonant cluster,
4. up to two vowels in the last vowel cluster,
5. up to three consonants in the final consonant cluster.

Empty positions are padded with empty phonemes, excessive phonemes (not corresponding to any of the positions mentioned) are discarded. Now, phonemes are not features. Each phoneme is encoded by a vector of 10 features such as `[voice]`, taking as value present or absent. These feature vectors are appended to form the full 130-node surface feature vector encoding (supposedly relevant parts of) any input word.

These 130 nodes — along with the semantic cues — are connected to only 20 nodes in the first hidden layer, and also in all other respects model 3 is just like model 1 and 2. It goes to show that manually selected cues are of no advantage over a general description of the input, combined with a learning method (such as a network).

3.3.2.1 Comparison

It is difficult to compare MacWhinney et al.'s result to the pure gender assignment in section 3.2. Gender is irrelevant in plural; all three genders take the same articles, only depending on case. Case was sufficiently represented in the input nodes for the contexts. Also in the singular it is not always necessary to distinguish gender: In dative and genitive, masculine and neuter nouns coincide in choice of article. Assume that number and case are sufficiently encoded, so that the accuracy of the method only really depends on its ability to predict (or recall) gender. Then, model 1's recall of 92% and precision of 61%, and model 3's recall of 93,6% (no generalization to unseen words is given) could stand for their ability to predict gender. Note that recall is evaluated on seen words in unseen contexts, and that errors here are the result of training contexts for these words which did not reveal their gender. In these cases, recalling a seen word's gender is impossible. As I cannot estimate the impact of this problem, the recall rates are not easy to compare to my results.

For precision, there is only model 1, and 61% do not impress. Yet the training data of only 102 nouns will not have sufficed to learn the influences of all cues. Note that the problem of how conflicts between cues are to be solved (cf. 3.3.1) was left for the network to sort out.

From my perspective a full comparison between a network with explicit input features and the KPCA method is not worthwhile, because 1. PCA arrives at a mathematically precise solution for perfect recall immediately, whereas MacWhinney et al.'s networks needed to be trained for at least 50 epochs and 2. a kernel has much more flexibility.

The use of explicit feature vectors as input to a network is also found in Plunkett and Nakisa (1997), Plunkett and Juola (1999), Hahn and Nakisa (2000) (see 2.1.4). Other models using explicit features (Naive discriminative learning, Baayen et al. 2011, see 2.1.5) need to deal with large, sparse feature vectors/matrices. In network approaches, each new input feature adds a new node and a weight to be trained. Furthermore, the method requires an elaborate pre-processing by aligning the features with the right slots. The model is thus very sensitive to the position of a feature, and will not generalize beyond that. This is known as the *alignment problem* (Bullinaria 1997). The implicit features of a kernel are not bound to a certain position. This is both an advantage and a disadvantage; to overcome the latter at least partially I introduced the edge-of-word sentinels.

It is possible to make a kernel sensitive to the positions of its implicit features, by considering two features identical only if they also appear at the same place. This place may be defined relatively (see word edges) or in absolute terms, that is, bound to a certain input node. Any explicit feature vector can immediately be used in a kernel method, by taking vector products as the kernel. Kernels are a generalization of explicit feature vectors. And with methods such as PCA or Naive discriminative learning (see section 2.1.5) one achieves the same as with a network — mapping inputs to outputs — in a mathematically precise way. In this respect, they can take the place of connectionist networks.

3.3.3 Support Vector Machines

Support Vector Machines (SVMs) are the standard way to classify with kernels (see 2.3.2 and Cristianini and Shawe-Taylor 2000). Nastase and Popescu (2009) use it to predict gender in German and Romanian. They achieve 72.36% in 10-fold cross-validation on a training set of 1731 nouns. This figure stays behind the results presented here for several reasons.

The nouns are encoded as concatenations of bundles of 28 binary phonetic features, more than in other approaches (16 for Hahn and Nakisa 2000; etc.). Segments are not assigned to slots, because this is not necessary in combination with a kernel. In contrast, the explicit binarization is only needed in network models. It is a central claim of this thesis that kernels overcome the need for this. Nastase and Popescu concatenate the explicit features into a string and use an (up-to-)n-gram kernel on this. This appears to me as a very naive application of a kernel.

It remains unclear to me whether they put an upper bound on the substring length n. This would have to be high, because even a single phoneme spans 28 positions in the input representations. Phoneme boundaries are not marked, only word edges (and leaving them out reportedly decreases the performance to 66.91%). Considering this, the kernel counts many shared subsequences aligned

at different places within a phoneme (or across phoneme boundaries). In these subsequences, the individual 1s and 0s denote completely different phonetic features. They are thus not identical at all and should not count towards the kernel value.

This artificially boosts the number of shared features, which leads to an exceptionally peaked kernel: Very high self-similarities as opposed to kernel values between non-identical items. So Nastase and Popescu normalize the kernel value (see appendix A), as a way to overcome this. Normalized kernels however do not allow to restore hidden feature counts. In preliminary experiments I found that this makes them unsuitable for feature inference.

In conclusion it seems that this implementation of SVMs to gender classification does not make proper use of kernels, mostly in that it counts entirely unrelated substrings towards similarity. The performance of 72% then shows that even naive kernels will learn quite a bit. Maybe this is due to accidentally shared features occurring evenly distributed.

3.4 Conclusion

Data-oriented classification, that has been traditionally done with neural networks, can nowadays be abstracted over by several methods. Naive discriminative learning, as described in 2.1.5, directly calculates the weighting matrix from surface features to hidden features. There is no training of weights involved. It also abstracts over the layer of exemplars (words) stored in memory. The other method is the KPCA model presented here. It emphasizes the word layer, and abstracts over the surface feature layer. Both models are useful in classification / identification of hidden features. With kernels, surface similarity can be defined more flexibly, which holds some promise for future advancements. The model can be applied to any classification task based on analogy, by defining a measure of relevant surface similarity (the kernel).

As shown in 3.2, a KPCA model of German nouns successfully predicts gender, in as far as gender is predictable. It can learn phonological and morphological cues, given enough data, but it does not (directly) exhibit a default. This is where it falls behind a rule-based model (3.3.1). Its advantages are that it can be built automatically, that it can learn any exception parsimoniously (that is, with at most one more principal component), and that it will improve if given more data.

It outperforms the only other kernel-based model of German gender, the SVM model of Nastase and Popescu (2009, see 3.3.3), however mostly due to naive design decisions in the latter. There was no useful interpretation to the principal components; this is addressed in the next chapter.

The dimensions of maximal variation within the German noun data highly correlate with the morphemes `-ung` and `-tion` (3.2.2.1). This is not to say that KPCA could used to identify morphemes: It is simply the high frequency of these morphemes that make it reasonable to first divide the data set by asking whether a word contains one or the other morpheme. The frequency in turn is an artifact of the corpus. If the corpus was balanced, or represented token frequencies, the result would be different.

Also note that the components only seem to denote these morphemes: They also contain arbitrary

features which happen to be correlated. In the component for `-tion`, `a` (and `atio`, and so on) is contained to 50%, because many words ending in `-tion` actually end in `ation`. Also the essential features of `tion` (`tio`, ...) are contained to various degrees. The component is not only used for this morpheme, but everywhere these features appear. It is a pure artifact of the data set.

Chapter 4

Interpretation of the factors

This chapter is about interpreting the model, following up the observations in 3.2.2.1. The interpretation of a principal component (2.3.4.3) hinges on the criteria defining its extraction: orthogonal to all others, and capturing the maximum of remaining surface variation. This leads to two methods, which are discussed in this chapter: Reduction of the set of principal components to the most relevant ones (cf. 2.3.4.2), and rotation of the principal components with respect to alternative criteria.

Principal component analysis is often used to reduce the dimension of a data set. This is done by discarding the principal components with lowest eigenvalue. Depending on the type of data, these are considered noise or otherwise less relevant information. In the case of the CELEX database, and the model developed in 4.4, annotation errors and exceptional patterns are hard to tell apart (due to the partial arbitrariness). Components capturing these are important for perfect recall, but a pattern instantiated by only a single word need not be part of the generalizing model. In section 4.1 I demonstrate that in fact the predictive power of the model is contained in the higher principal components, and that a lot of the less relevant ones can be discarded. Instead of describing the data by itself, storing each and every exemplar, the data points are now represented by a few concepts, the remaining, information-bearing basis vectors. Thus we arrive at a conceptually smaller model.[1]

However note that the principal components are ordered by their explanatory value towards surface forms, not towards gender. Now a large variation in surface form might very well be correlated with gender variation, but this is not necessarily the case. Low-ranked principal components might cover large gender variation, so discarding them results in a worse model of gender. Furthermore, they represent just one out of infinitely many bases of the feature space: Other vectors within that space might give far better gender prediction. PCA as a feature extraction method does not identify linguistically meaningful features from the set of German nouns, and this would be true for many similar linguistic spaces.

The linguist has already identified and annotated gender as a hidden feature. The goal of this chapter is to identify components related to gender. There are three different values for gender, so in order to keep them apart, two dimensions are needed at least. That two dimensions do indeed suffice

[1] In terms of storage space, the representation is larger: Each vector is given as a linear combination of the data items, so not only the whole data set needs to be stored, but also the weighting factors.

will be shown later (4.4). The means for identifying these dimensions is rotation (4.2). The original and popular VARIMAX (Kaiser 1958) rotation for non-kernelized PCA does not apply to Kernel PCA (4.2.2). I will formulate a rotation criterion which leads to very few maximally gender-loaded components (4.2.3), rendering the rest neutral towards gender.

Along these two components, all gender variation can be depicted. Knowing the similarity of a new item towards all training data (its kernel vector), these are the only two dimensions to be looked at in order to predict its gender (to the extent possible). So these dimensions are *extractors of hidden features*. Necessarily, training items of the same gender are indistinguishable within the plane spanned by those two dimensions (4.3). Unseen testing items will fall anywhere in the plane, forming clusters around the three exact gender points. Clusters will overlap due to gender assignment errors and gender arbitrariness. However, all mis-classified data can be incorporated into a new model, thus giving all those data items perfect recall.

The two dimensions of gender variation are maximally informative regarding gender, capturing all gender variation within the data and retaining the full predictive power of the model. The method presented here provides gender extractors which are practical, and as few and as precise as they can be. Still, they are not linguistically interpretable, because of what they are made for: If they are to perfectly recall the training data, then they will be most over-fitted to the particularities of the data. They do not define abstract rules of gender assignment. Rather they are very concrete, data-oriented descriptions.

This chapter is structured as follows: Section 4.1 shows how much of the relevant information is contained in subsets of varying size of the most explanatory dimensions of variation (the principal components with highest eigenvalue). In section 2.3.4.3 I argue that principal components generally are not interpretable as features of the data because of their definition of variance. This problem is usually tackled by rotating the principal components. Section 4.2 discusses VARIMAX rotation (4.2.2), and argues for another rotation (4.2.3) motivated by a different criterion of variance. As usual, I give a detailed walkthrough using the example from 3.1 in 4.3. Section 4.4 then gives the results for the large data set. In section 4.5 I show the limits of factor rotations: Even under ideal conditions, they cannot extract any knowledge (meaningful factors) beyond what was previously known. The final discussion is in 4.6.

4.1 Dimensionality reduction

So far, I employed all principal components at the same time, so the model did not benefit from their ordering. In the following, I first show how cutting off the principal components affects the model's accuracy. 20% of the components restore the gender assignment of 80% of the data items. Then, I will show that the ordering of components is actually ignorant of gender, since the kernel did not know anything about it (section 3.1). Other components might capture gender variation far better than this. In fact, I will rotate the extracted principal components so that only a very few of them (in fact, only two) are related to gender, while the others are not. This is done by a variant of VARIMAX

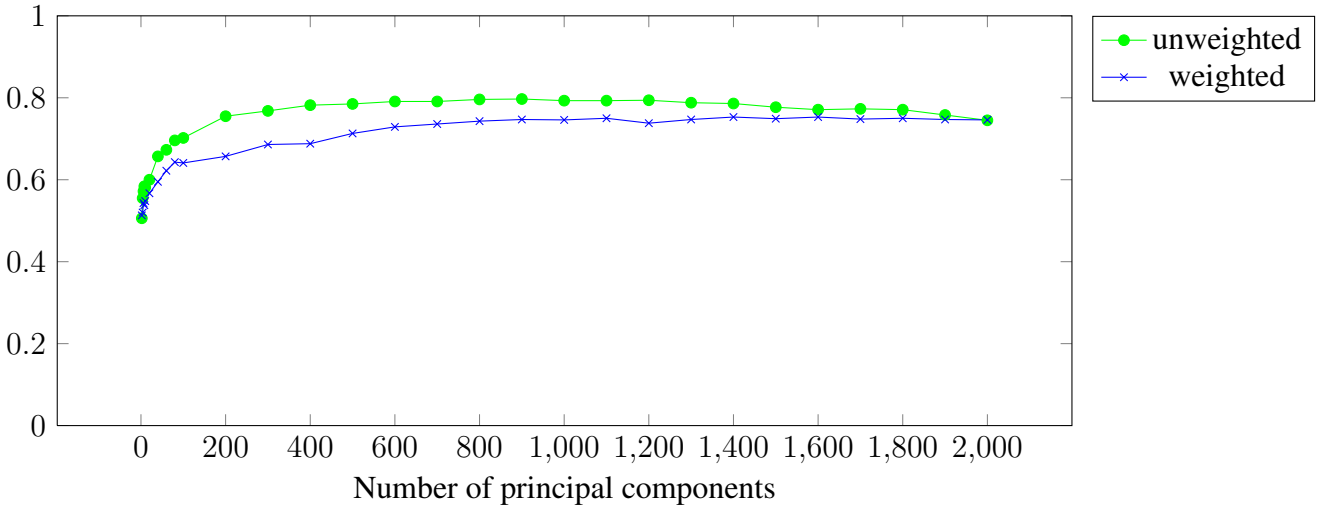


Figure 4.1: Gender assignment accuracy

(4.2.2).

4.1.1 Experiment

From the experiment in 3.2, I pick out a representative setting (4-gram kernel on the orthography, 2000 data items), and show the effect of reducing the number of dimensions in figure 4.1. The results are averages over 10 runs with different, randomly selected data. I compare KPCA to Weighted KPCA (2.4.2). The assumption is that the first few principal components of WKPCA would contain more important information than the same amount of components from unweighted KPCA. Several observations are to be made:

- A prediction is counted as correct if the correct gender received a higher value than the two others (simple majority). Therefore results are robust to the loss of some information: Only after discarding a certain number of principal components, the outcome for a given word will change, if at all.
- There are usually around 1994 dimensions, due to redundancy (as seen from the kernel's limited n-grams) in the data.
- Discarding half of the principal components (those with lowest explanatory value) actually leads to a slight increase in precision. These components do not capture general tendencies, only particularities of certain data items. These are not needed in generalizing to unseen data. As it turns out, it may even be harmful because of over-fitting.
- With only the 200 principal components of highest eigenvalue, the unweighted model is still at peak performance, that is, at least as good as with all principal components.

- With fewer and fewer components, the model approaches its baseline.
- The weighted model coincides with the unweighted in the case of full rank, and it approaches the same baseline. Everywhere else it performs worse.

4.1.2 Discussion

The performance of the method hardly increases beyond 1000 items of training data (see 3.2). At 2000 items, 200 principal components still capture nearly all its predictive power. I assume 200 principal components will be enough even for a larger data set. This benefit of PCA, the filtering of dimensions by variance, has been put to good use here. As we are going to see, all gender variation actually falls into a two-dimensional subspace (4.4), but it is far easier to extract these two dimensions by rotating 400, not 2000 dimensions (4.4.2).

Weighted Kernel Principal Component Analysis did not hold up to expectations. It appears to be the case that a model actually benefits from treating all data alike, such that it can cover more cases. Relying more on the more frequent cases will lead a model to favor the most frequent cases, and consequently it will be at a disadvantage for the rarer patterns. After all, exemplar models are about explaining sub- and irregularities. One might say that the WKPCA model is overtrained. As mentioned in connection with exemplar models (2.1.1), frequent classes rather need to be *discounted* than boosted. This is contrary to what WKPCA does.

This concludes my discussion of WKPCA. There might be other domains of application where its frequency-sensitive properties actually lead to a model closer to human performance. These results are not to say that human judgments do not exhibit frequency effects; but it does suggest that humans (here: native speakers of German in guessing the gender of a noun) do make use of infrequent patterns, and are not influenced by different patterns in direct proportion to their frequencies.

4.2 Rotations

As seen earlier (2.3.4.3, 3.1.2 and 3.2.2.1), the principal components are hard to interpret. In particular, they have no linguistic meaning, they just denote dimensions along which the data is spread most widely. Another basis of the same feature space may be more useful. This basis retains orthogonality and the feature space spanned.

4.2.1 Preliminaries

Principal components are chosen to maximize a certain criterion of variation: Each component, in turn, captures the maximum of the remaining variation (cf. 2.3.4). This notion does not lead to easily interpretable components. A component is a dimension in feature space, given as a vector from the origin to a certain point. This point is expressed as a weighted combination of (images of) data items. Typically, every item only plays a minor role in expressing the component. This means that PCA does not extract prototypes, nor are the principal components exemplified by only a few exemplars.

The flip side of expressing components with data items is expressing data items with components: Locating a data item in feature space. That is, characterizing it by the values it takes on each basis vector. Again, it will typically *not* be just a few factors with high values and all the others near zero. Therefore the notion of a *component* is misleading. Data items are not *composed* of a few components (possibly even in whole numbers; also see 4.5). Small items do not need less components to be characterized. Those descriptions are not parsimonious at all. Rather, principal components are correction terms: Starting from the mean of the data, they approximate the item component by component.

Latent semantic analysis (LSA; Deerwester et al. 1990; see also a similar variant with Independent Component Analysis, ICA, Honkela et al. 2010) achieves recognizably similar descriptions for similar items. High values or low values on some specific subset of the factors characterize a cluster of items. One finds that some factors are correlated with certain semantic classes, but they do not denote these classes. Results are often interpreted just by manual inspection (see 2.3.5.1).

This uninterpretability of principal components is due to their aim: They are axes of maximal variation, in the high-dimensional cloud of data item images. They do not necessarily point to clusters within the data, they are two-fold distinctions between groups of clusters. They are nothing like what the researcher would have posited as 'hidden features' in the data.

Two properties of the factors need to be retained: Their independence (orthogonality), and the space they span. Now any rotation of an orthonormal basis still spans the same space. A step in increasing interpretability of principal components is rotating them, according to another criterion of variance — not the one maximized by PCA. The standard way to do that is VARIMAX (Kaiser 1958), an algorithm for variation maximization. Other rotations, some of which generalize to non-orthogonal solutions, are beyond the scope of this thesis.

This chapter proceeds as follows: I argue that classical variation maximization cannot provide more interpretable components (4.2.2), in this case: components which are informative with respect to gender. I then reformulate the variation criterion (4.2.3), such that it will maximize the variance of gender loadings within the components. By an algorithm very similar to VARIMAX, the orthonormal basis obtained by the PCA on the German noun data can be rotated such that two components span the plane in which all gender variation lies. All other components will be orthogonal to that, and thus totally unrelated to gender. Then, in order to predict gender, only these two components have to be looked at. The precision remains the same, compared to the model with full principal components.

4.2.2 VARIMAX rotation

The main incentive for a factor analysis such as PCA is to reduce the wealth of surface features to a fewer number of independent factors. VARIMAX addresses the interpretability of these factors. The intended interpretation is such that each factor can be described in terms of a combination of a few surface features, bearing a high loading each. All other features then hopefully have loadings close to zero. Then the factor denotes a bundle of features, and this is in fact human-readable, and maybe even linguistically meaningful. The definition and an algorithm are found in appendix E.1.

The criterion of variance used in VARIMAX is the sum over the variance of (squared) feature loadings within each component. So components with largely diverging loadings are preferred: at best, they have some very highly loaded features and zero loadings elsewhere. The general algorithm proceeds by rotating principal components in pairs (within the plane they span), until no rotation of any pair can further increase the criterion.

To my knowledge, the particularities of applying VARIMAX to the kernelized variant of PCA (that is, Kernel PCA) have not been discussed in the literature.[2] Rotation and kernels exclude each other, due to the variance criterion being defined on explicit surface feature loadings. Principal components are not expressed in terms of feature loadings, but as item loadings. One could maximize these, with the intention of obtaining prototype-like factors. The problem is that the data items (words) are larger than the features one might want to extract (relevant properties/parts of words). These features cannot be expressed as prototypes. Thus VARIMAX by items is not what I want.[3]

There is a third type of loading that directly leads to useful components. As said earlier (2.3.4.3) and as will be shown in the walkthrough (4.3), the principal components of the space of German nouns are unsystematically loaded for gender, because this was not accessible to the kernel. It is the variance of these loadings that needs to be maximized, as explained next.

4.2.3 Rotation towards features

I now present a rotation method closely related to VARIMAX, yet operating neither on explicit surface features nor on item loadings, but on loadings of hidden features. This *rotation towards features* will result in components interpretable with respect to those (and only those) features.

Every component is loaded for each gender. These loadings are obtained by multiplying the components (represented as linear combinations of training items) by the hidden feature matrix (see appendix B, definition 26). This takes items to their hidden features, and thus combinations of items to loadings of features. We get a components-by-hidden-features loading matrix.

As in VARIMAX, loadings are now squared, and the variance within *features* is calculated: features shall be highly present in a few components, and near zero in all others. These variances are summed over all components. This opposes to a *component*-wise calculation, where each component shall have a few distinctive features, as in VARIMAX. The difference is due to the relative numbers of factors and features. One usually has more surface features than factors, and therefore each factor can pick its features (VARIMAX). On the other hand, there are comparably few hidden features, at least in the scenarios I envision. Thus the hidden features pick their factors (rotation towards hidden features).

The algorithm shown in appendix E.2 now rotates the basis given by the principal components such that this sum becomes maximal. It does so by locally maximizing the sum of variances for two components, pair by pair, and starting over at the beginning as long as the score improves. This

[2] The text books (Schölkopf and Smola 2002, Shawe-Taylor and Cristianini 2004) do not mention rotations. I would be grateful for any relevant reference.

[3] A similar argumentation applies to ICA: It is a rotation of the components of an SVD, and it requires explicit features. Thus it cannot be directly combined with a kernel.

method directly borrows from VARIMAX. The derivation of the optimal rotation vector for each pair of components is also given in the appendix; it is a matter of setting up the appropriate equation and solving it, and therefore of no explanatory value here.

It turns out (see the application in 4.3 and 4.4) that it is optimal to concentrate all variance onto as few components as possible.

4.2.4 Reducing the dimension

If one is no longer interested in surface variation beyond what is needed to explain the hidden features, one can discard all components with zero hidden feature loadings, and only retain the dimensions of hidden feature variation. The number of remaining dimensions is bounded by the number of hidden features (minus one). Data items (both seen and unseen) projected onto the remaining dimensions will completely retain their hidden features. These dimensions span the space of all variation within the hidden features.

Data items which are indistinguishable with regard to their hidden features have to occupy the same point in this reduced data space. Within such a point, there is no surface variation at all; and only a small amount of surface variation is captured in the reduced space. The remaining surface variation is captured by the other dimensions, which can be discarded for the purposes of extracting the hidden features.

4.3 Walkthrough

This section demonstrates the rotation of principal component towards hidden features, using the example from the last walkthrough (3.1). 9 training items and a bigram kernel yield (in general) 8 principal components. However, they are initially blind to gender. Therefore they are rotated in order to maximize the variance of their gender loadings. The gender loadings of the components 4.1 are obtained by multiplying them (3.5) by the matrix of hidden features 3.2.

(4.1) Gender loadings of the original principal components

data item	pc1	pc2	pc3	pc4	pc5	pc6	pc7	pc8
masculine	-0.205	-0.054	-0.033	-0.482	-0.125	0.086	0.484	0.582
feminine	0.097	-0.110	0.149	0.322	-0.331	0.049	-0.523	-0.091
neuter	0.107	0.164	-0.116	0.159	0.456	-0.136	0.038	-0.491

Every component is loaded for gender, in an unsystematic way. The components do not tell anything about gender specifically. A better distribution would be one where few components have high gender loadings and all others only low loadings. This is captured by the revised variance criterion (4.2.3; calculates as 0.032 here), and the revised rotation algorithm (appendix E.2) finds the following two vectors in feature space, describing the plane of *all* variation in gender (4.2).

(4.2) Gender loadings after rotation

	pc1′	pc2′
masculine	-0.003	-0.936
feminine	0.572	0.463
neuter	-0.569	0.472

The solution found by the rotation algorithm is one where one component (pc1′) separates feminine and neuter, and another (pc2′) distinguishes masculine from the two. Remember that masculine was the predominant gender in the data set (5 out of 9). The new vectors are nearly symmetric; that is, they have two loadings with the same absolute value. Note that the sum of gender contributions by those vectors is 0. This is because the data was centered, that is, the mean was subtracted. The mean gender is [0.555 m, 0.222 f, 0.222 n], with a sum of exactly 1 gender feature per data item. There is no variation with respect to the sum of the gender values in the data. Therefore no vector can change the sum. The origin of the principal components is not the null item, but the mean item. On the vector describing the data mean every item is forced to take the value 1. Otherwise they could describe words with several gender values, or none at all.

All other dimensions are neutral towards gender, i.e. they have 0 gender loadings. The gender of a noun (from the training set) never depends on the value it takes on one of these dimensions. Instead, its gender is fully described in only two dimensions. These two dimensions describe the plane of gender variation. The variance of gender loadings now is 0.113.

Seen data items have perfect recognition. They will be mapped to one of three points within the plane; see 4.2. Unseen data items may lie anywhere, although the better-than-chance performance of gender prediction (see the full experiment in 4.4) requires that they will not be randomly distributed. See also figure 4.3.

(4.3) First two rotated principal components

data item	pc1′	pc2′
`blau`	-0.210	0.036
`traum`	0.149	-0.137
`tau`	-0.358	0.436
`baum`	0.139	-0.061
`raum`	-0.265	-0.026
`sau`	0.324	0.342
`frau`	0.248	0.121
`pfau`	-0.030	-0.226
`stau`	0.002	-0.485

All basis vectors have been rotated; all have changed their gender loadings. Thus they no longer describe the dimensions of maximal variation in the input data. Their order and the eigenvalues become meaningless. Their numbering is unrelated to the the numbering of the original components: It is not necessarily the case that pc1′ is obtained from pc1, and so on.

The two dimensions of gender variation are orthogonal with respect to gender, as their product shows. This is in addition to their already established orthogonality by items (see also 4.3) and by implicit features (via Kernel PCA).

When determining gender, one can now safely disregard all unrelated dimensions. The data items are mapped onto their two gender-bearing dimensions as follows:

(4.4) Neuter: `#blau#`, `#tau#`

Coordinates: [-0.8707, 0.5967]
Gender loadings: [-0.555 m, -0.222 f , 0.777 n]
plus mean: [0, 0, 1]

Features:
$-0.181\times$ `st`
$-0.148\times$ `#s`
$-0.148\times$ `s`
$-0.101\times$ `tr`
$0.178\times$ `um, m, m#`
$0.192\times$ `ra, r`
$0.268\times$ `#b`
$0.268\times$ `b`
$0.316\times$ `la, bl, l`
$0.326\times$ `#r`
$0.401\times$ `t`
$0.502\times$ `ta`
$0.582\times$ `#t`
$0.821\times$ `u#`

(4.5) Masculine: `#traum#`, and all others

Coordinates: [-0.0038, -0.4745]
Gender loadings: [0.444 m, -0.222 f , -0.222 n]
plus mean: [1, 0, 0]

Features:
$0.124\times$ `#r`
$0.139\times$ `ba`
$0.175\times$ `tr`
$0.218\times$ `#p, p, pf, fa`
$0.234\times$ `#b, b`
$0.247\times$ `ta`
$0.270\times$ `f`
$0.289\times$ `#s, s`
$0.341\times$ `st`
$0.352\times$ `ra, r`
$0.422\times$ `t`
$0.439\times$ `um, m, m#`
$0.560\times$ `u#`

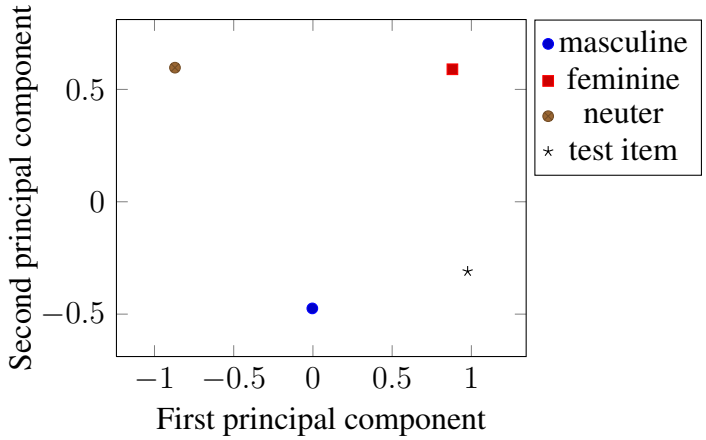


Figure 4.2: Feature space of the dummy data, after rotation

(4.6) Feminine: `#sau#`, and all others

Coordinates: [0.8803, 0.5896]
Gender loadings: [-0.555 m, 0.777 f , -0.222 n]
plus mean: [0, 1, 0]

Features:
$-0.172\times$ `st`
$-0.138\times$ `#r`
$-0.120\times$ `ta`
$0.145\times$ `#b, b`
$0.161\times$ `tr`
$0.198\times$ `ba`
$0.214\times$ `#t`
$0.221\times$ `um, m, m#`
$0.352\times$ `f`
$0.401\times$ `#f, fr`
$0.425\times$ `ra, r`
$0.425\times$ `#s, s`
$0.598\times$ `sa`
$0.778\times$ `u#`

All nouns of the same gender are mapped onto the same representation. As seen from their gender feature, there is is simply no distinction. The three points given are not really prototypes of the respective gender. They are not even averages, because averages would be correlated. The surface features described by these points are merely by-products of the necessary requirements (orthogonality of the basis, perfect recall of the training data). All seen words of a given gender digress from that point in many ways, namely in the 6 remaining dimensions, which all are orthogonal to the plane of gender variation.

These representations are not only the contributions by the two dimensions; they are a full reconstruction restricted to those two dimensions. In practice that only means the mean has already been added to them. From the lists, I omitted the ubiquitous `a`, `u`, `au` (all 1.0) and the sentinels `#` (2.0).

The test item, `#saum#`, is placed according to its inferred gender. The two gender-bearing dimensions assign to it the very same distribution over the three genders as the 8 initial components did. It is far away from neuter, and a bit closer to masculine than to feminine. Compared to fig. 3.2, it is much easier to see these relations in fig. 4.2, because each gender forms a maximally tight cluster, and is not distributed all over the first two dimensions.

Summary

The initial 8 principal components are arbitrarily loaded for gender, since gender was unavailable to the kernel. When supplied with gender information, the basis vectors can be rotated in order to maximize the variance within their gender loadings. The variance criterion is formulated such that a few components will grab all the gender variation, while the others become neutral towards gender. After rotation, there are exactly 2 dimensions spanning the plane of gender variation, and 6 remaining dimensions taking care of surface variation. All seen data items are mapped onto one of three points in that plane; one for each gender. The unseen item is mapped such that all relative similarity is preserved.

4.4 The two-dimensional variation of German gender

I now apply rotation towards gender-identifying dimensions to a larger data set. Following the practice in 4.1.1, I use 2000 training items and a 4-gram kernel, comparing the orthographic and phonological input representations. First, a Kernel PCA is performed as in the gender prediction experiment (3.2). Then, the dimensions are rotated as described in 4.2.3 in order to identify the plane of gender variation. The model is tested on 100 unseen items before and after rotation. As observed before, the precision stays the same, because the extracted gender loadings are the same, whether it is with all dimensions or just the two capturing all gender variation.

4.4.1 Results

4.4.1.1 Orthography

The experiment on orthographic data finds 1990 dimensions of variation in the space of German singular nouns. The variance of gender loadings as calculated by the new criterion is very low, but can be maximized by rotation. 1988 of the new basis vectors have zero gender variation, because they are neutral towards gender. The other two have very high variation, as shown in 4.7. The exact values do not matter. The model predicted 75% of the unseen test items correctly. Figure 4.3 shows the clustering visually: Each gender is centered around its exact point as is needed for the particular training set. 25% of the points lie too far off, and are falsely recognized as members of another class.

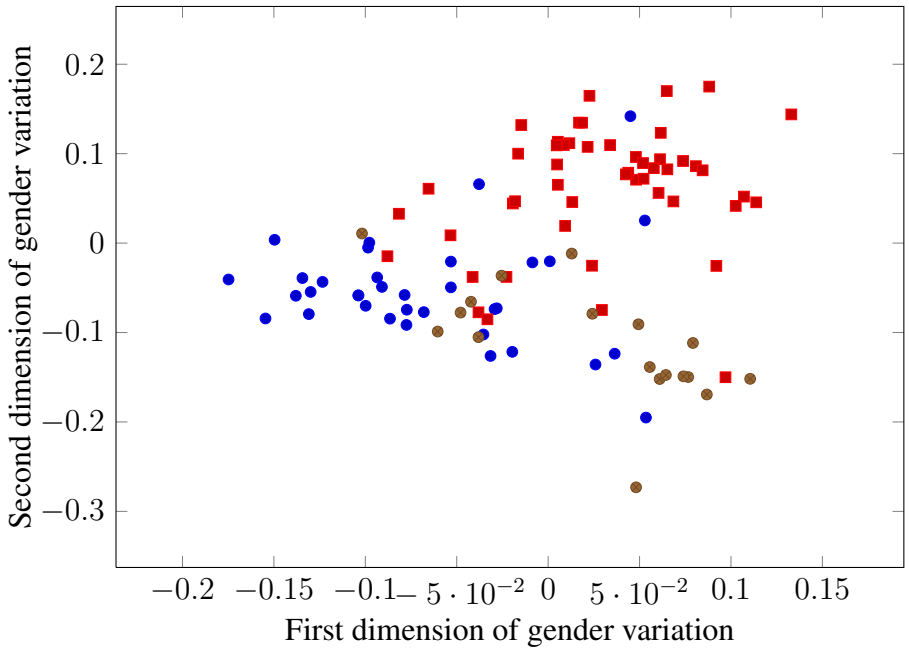


Figure 4.3: Orthographical gender clusters

Dimensions: 1990

Variance of gender loadings: 0.0036

Variance after rotation: 0.9717

Precision: 0.75

(4.7) Gender loadings after rotation, orthographic

	pc1′	pc2′
masculine	-5.8880	-0.6848
feminine	2.1833	4.3179
neuter	3.7046	-3.6331

4.4.1.2 Transcription

For the experiment on phonological transcriptions, results are very similar; see the visualization in 4.4.

Dimensions: 1991

Variance of gender loadings: 0.0021

Variance after rotation: 0.7213

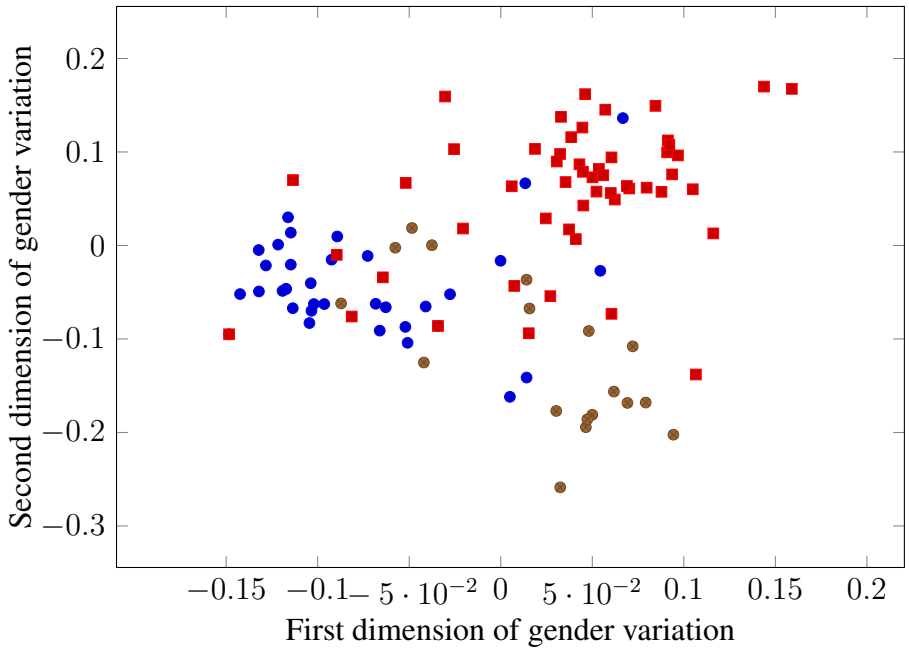


Figure 4.4: Phonological gender clusters

Precision: 0.78

(4.8) Gender loadings after rotation, phonological

	pc1′	pc2′
masculine	-5.4800	-0.1820
feminine	2.5432	3.8923
neuter	2.9368	-3.7103

The main conclusion is that there is in fact a two-dimensional subspace of the space of German nouns (as induced by a kernel over their surface forms) describing all gender variation, and that the the three genders are not randomly distributed in that plane. They form clusters around center points, which each represent *all* seen instances of one gender. The clusters overlap, leading to the misclassification of some of the unknown nouns. Reasons for mis-classifications are discussed in 3.2.2. An upper bound for prediction is given by the informativeness of gender: If it were perfectly predictable, then it would carry no information. Yet it is an integral property of German nouns.

There is no significant difference between the two data representations (orthography and phonology). The extracted dimensions of gender variation are very similar (see 4.4 and 4.3). This is to be expected, given the correlation between the two annotation layers.

4.4.2 Discussion

The successful reduction to just 2 dimensions in the case of gender – placing the 3 values in a plane — hinges on the implementation of the algorithm (see appendix E), but given enough precision it

was always achieved in my experiments. This is also true in combination with reduced dimension (4.1): When first reducing the number of dimensions, the rotation will still identify exactly two dimensions capturing all the gender variation within the lower-dimensional subspace of the kernel-induced feature space. Given that the rotation algorithm is computationally costly, and that the number of pairwise rotations grows quadratically with the number of principal components, reducing the principal components to start with is an effective and legitimate way to speed up the computation. Following 4.1, 400 dimensions are enough for the given task.

The resulting plane of gender variation highly depends on the training set, since it is meant to exactly represent all training items. A more general model is obtained by first reducing the dimension. Still the components are arbitrary in the sense that they can be rotated within the plane. Their item and feature loadings are such that the component does what it does; there is no immediate interpretation. This is to say that this model does not extract the 'rules' of German gender. It is only a perfect (100% accuracy on training data) abstraction over the training data capable of 80% correct prediction. They can assign gender to an unseen word based on its similarities to all known words. They do not provide a human-readable explanation of how they do it.

It is even the case that similarity to a feminine word can hint towards neuter: The data set contained `zession` (*cession*), which is a strong indicator for neuter, probably because of its beginning `#ze` (as in "Zelt", n., *tent*). There are enough words ending in `ion#` such that even the word `zession` itself will be mapped to the feminine center point. This shows that the composition of the new dimensions is not even necessarily predictable by gender itself. Quite the contrary, it is very arbitrary and specific to the training data.

4.5 Uninterpretability of factors

In this section I discuss a few simple examples with respect to the interpretability of factors. The point is to show what can be done at all with rotation towards features (4.2.3).

4.5.1 A very simple example

Consider the following language (a set of strings of fixed length):

(4.9)
$$L_1 = \{a, b\}^6$$

A unigram KPCA (with 1000 items) derives a single principal component, which models the ratio between the two features.

(4.10)

feature	pc 1
a	-0.725
b	0.689
eigenvalue	3176.5

The values of the feature loadings derive from normalization: The sum of their squares equals 1. Their absolute values are not equal because of the random sampling of the language. PCA is sensitive to frequencies, and therefore it cannot conclude that a and b should be treated alike.

In the next language, I relax the length constraint.

(4.11)
$$L_2 = \{a, b\}^i, 0 < i \leq 12$$

Again on 1000 items, a unigram KPCA finds two factors:

(4.12)

feature	pc 1	pc 2
a	0.679	0.730
b	0.748	-0.682
eigenvalue	5880.7	3261.7

The first principal component adds both as and bs. Thus it sorts items by size. It is slightly off being symmetric, because of the random sample of data items. The second component serves the same purpose as the one from the previous example (4.10)

Intuitively, the items are composed of as and bs, and not of a size parameter and a ratio parameter, although this is precisely what KPCA thinks is the best way to explain the data. The data falls into a plane with is spanned by the features a and b. Any pair of orthogonal vectors in that plane is a valid basis, e.g. the ones given above (4.12). So rotations might reveal a more interpretable basis. VARIMAX by items only adjusts the vectors minimally. Their interpretation stays the same.

Rotation towards features needs explicit features. Let the symbols of the alphabet be the 'hidden' features,[4] and B the items-by-alphabet matrix of hidden feature counts. Then $B_{i,j}$ denotes the occurrence count of the jth symbol in item i. The rotation now enforces one factor for a and one for b, because in the constellation below (4.13) the loadings are maximally variant within a feature.[5] Thus KPCA with rotation manages to find the trivial solution.

(4.13)

feature	pc 1	pc 2
a	1.002	0.000
b	0.000	1.005

4.5.2 A not so simple example

In the last example, surface and hidden features were identical. This led to the circular conclusion that the (surface) features are the (hidden, or extracted) features. The situation becomes less clear if we add information to the kernel. Nothing changes about the nature of the data when I use a bigram kernel (not including unigrams), but the principal components do change.

[4] They are both surface and hidden features now.

[5] Here this coincides with the loadings also being maximally variant within each *factor*, but this is not what the algorithm maximizes.

(4.14)

feature	pc 1	pc 2	pc 3	pc 4
aa	-0.407	-0.767	-0.482	0.015
ab	-0.516	-0.039	0.473	-0.715
ba	-0.512	-0.017	0.502	0.698
bb	-0.549	0.638	-0.538	0.000
eigenvalue	2966.2	2450.7	1259.2	233.9

The first factor denotes the size of the item, because all features have the same sign.[6] The following factors model the ratios between the surface features. This is not much different from the situation with unigrams. VARIMAX (by items) applies similarly and does not improve the interpretability. After rotation towards the explicit features `a` and `b`, two factors remain loaded for them, defining the plane of `a`-to-`b` ratios (cf. the unigram KPCA, 4.5.1).

(4.15)

feature	factor 1	factor 2
aa	-0.856	0.147
ab	-0.345	-0.359
ba	-0.335	-0.369
bb	0.145	-0.844
a	-1.248	-0.162
b	-0.156	-1.263

Note that while the two factors are of course orthogonal in terms of their bigram features, this is not true with respect to the unigrams. Also, the secondary feature of a factor does not reduce to 0. This is similar when including unigrams in the kernel.

In this last example the relevant features (unigrams) were specified explicitly. A final option is to include all surface features of the up-to-2-gram kernel, and rotate towards them.

(4.16)

feature	factor 1	factor 2	factor 3	factor 4	factor 5
aa	-0.408	-0.408	-0.408	-0.408	0.408
ab	-0.118	-0.118	-0.118	0.881	0.118
ba	-0.118	-0.118	0.881	-0.118	0.118
bb	-0.118	-0.118	-0.118	-0.118	-0.881
a	0.118	0.881	0.118	0.118	-0.118
b	-0.881	0.118	0.118	0.118	-0.118

This result is unsatisfactory. In no way it shows the true nature of the data: that it is composed of two atomic elements. If at least the dimension is known beforehand, the principal components can be restricted before rotation.

[6] The orientation of components is arbitrary.

(4.17)

feature	factor 1	factor 2
aa	0.532	-0.121
ab	0.201	0.205
ba	0.209	0.205
bb	-0.127	0.528
a	0.784	0.030
b	0.030	0.788

Looking only at the unigram loadings, this comes close to the desired result. In order to obtain it, one needs to know which or how many features underlie the data, and even then the result is only an approximation.

4.5.3 Discussion

Maximization of the variation of features does extract the subspace into which the relevant variation falls; yet it cannot guarantee that the basis vectors actually align perfectly with the features. In the examples above the model was hard pressed to find even the simplistic structure of the language over an alphabet of two letters (which supposedly is the alphabet itself). Arbitrary properties of the data (from random sampling) lead to variation. This is meaningful regarding the unigrams, but bigrams are only a by-product of the language definition above. Principal components describe exactly these arbitrary properties of the data. This variation is not noise, but genuine, so it is not filtered out by PCA.

There are other options to rotate the vectors, other criteria to maximize. I cannot exclude that some dimensions in a syntactic feature space are actually meaningful, but my findings advise against looking for them via PCA and (orthogonal) rotations. Another idea would be that building blocks simply are not orthogonal, but non-orthogonal rotations go well beyond this thesis.

4.6 Conclusion

In this chapter, I have first shown that the ordering of dimensions as given by KPCA can be exploited to generate a smaller model of the input space. Principal components with low eigenvalues explain only very little variance in the data, that is, only minor deviations of just a few items in the training data from what can be described with the more important dimensions. Discarding this highly data-specific information does not lead to an immediate drop in performance. Actually, even an improvement is possible due to the application of more general patterns.

The principal components are only unsystematically related to gender, since the kernel had no access to this hidden feature. They only describe surface variation. This is of no interest to theoretical linguistics. A standard trick to facilitate interpretation of principal components is to rotate them, e.g. with VARIMAX. For Kernel PCA, this would be variation by items, and this is not what one wants.

The method described here does not establish hidden features. It remains to be seen whether kernel-based methods can identify features without knowing them in advance. Presupposing them,

the principal components are loaded for them in various degrees. This is exploited as a new criterion of variance: The basis vectors are rotated such that a few of them cover all gender variation, while the rest is neutral towards gender. A vector/direction is neutral towards gender if the linear combinations of training items which describes it adds up to a zero value for each gender. The total sum of 0 is already guaranteed by the model: The mean introduces a total gender value sum of 1, and moving an item along the principal components can only change the proportions, not the fact that every word within the model has exactly one gender value.

This new rotation identifies the minimal necessary number of dimensions to describe the variation of hidden features. 3 different gender values can be placed into a plane. This plane is successfully and reliably identified. Words of one gender fall onto the same point in this plane. From this point, they inherit some of their surface features. In this respect, words of the same gender look identical. However, these features do not present meaningful surface properties for words of a certain gender. The other dimensions take care of the actual surface shape of the words.

This new two-dimensional model predicts the same gender values for all testing items as the full model. The testing items form loose clusters around the unique points for each gender. Thus two dimensions suffice to extract all information about the gender of an unseen word, to the best ability of the kernel-induced model.

Rotation towards abstract classes introduces an additional level of orthogonality. Consider the "naive" basis (4.3): It is orthogonal in terms of data items, and nothing else. KPCA finds a basis orthogonal in the space of implicit surface features. Crucially, it retains orthogonality by items. The rotation finds a new basis within feature space, such that the new basis vectors are orthogonal also by abstract classes. It retains the previously established orthogonality. Hence the final vectors are orthogonal by items, by surface features, and by abstract features.

As an alternative, consider KPCA with a kernel on gender. Let it be a simple identity kernel: 1 for identical gender, and 0 otherwise. This will also yield a two-dimensional plane, in which each gender occupies a unique point. This is so because as seen from the gender identity kernel, there is absolutely no way to distinguish words with identical gender. Then note that the two principal components of this KPCA cannot be orthogonal by surface features, since this level of description did not enter the model anywhere. This is analogous to gender being irrelevant in the original KPCA on surface representations. That means, a KPCA with a gender kernel cannot give the same result as KPCA plus rotation.

The ordering of principal components by data variance is lost in the process. As I argued in 2.3.4.3, there is no use for this in theoretical linguistics anyway. The ordering can however be used to reduce the dimension before rotation. This facilitates the process of rotation, and it reduces over-fitting of the model. As seen in 4.1.1, performance can improve with the discarding of low-ranked fine-tuning dimensions. Reducing the dimensional limits the predictive power, but no more is lost by rotation.

Over-fitting also means that any additional data can be accommodated in the model. It will always be specific to the training data. The shape of the components is a direct consequence of this, and therefore they will never encode a static (unchanged under additional data), abstract model of

gender. It is a live and active model of *performance*, solely driven by experience, as opposed to a model of *competence*.

Another idea to get some interpretation out of the two dimensions of gender variation would be to rotate them within the plane, according to a fourth criterion yet to be formulated, the three other criteria being the three orthogonality requirements already in place.

4.6.1 Outlook

Rotation towards hidden features can be applied to the abundance of abstract classes and grammatical features employed in the formal description of language (see 1.2.1). It identifies the relevant subspace of variation within the hidden features, as opposed to the full space of surface variation. Yet the hidden features have to be pre-defined by the researcher; they are not suggested by the model.

The method for rotation is iterative. It slowly converges towards optimal vectors. There might be a way to obtain a stable state directly from the criterion of variance, by finding an optimal solution analytically. This I have to leave to a mathematician. As the overall method now contains an iterative step, it loses some of its appeal as being exact, in comparison to networks, which have to be trained iteratively.

Rotations towards other criteria are conceivable. Some rotation methods find oblique solutions, that is, they do not insist on the orthogonality of hidden features. Such a model would then be free to posit any linear combination of training items as a hidden feature. In the current case, there could be a solution with one dimension per gender class. As they span a plane, each of them could then be expressed by the other two.

Chapter 5

Markup

In this chapter, a feature space model of a set of words is used to map unannotated word forms onto the corresponding annotations (specifically stress marks and syllable boundaries). The need to find concrete outcomes for the fuzzy representations returned by the method introduces *pre-imaging* methods.

5.1 Introduction

In inferring abstract features, a model maps from the data points into a small set. In the end, the method shall be flexible enough to allow the same complexity in the co-domain as in the domain. This chapter takes a little step in that direction, introducing the relevant concepts. A trivial case of equal complexity is a mapping from words to themselves. Nothing is to be learned from that. By adding a little markup, such as stress marks and syllable boundaries, to a word, the co-domain is still very similar to the domain. Every word is mapped to a copy of itself, with just a few symbols introduced.

In this chapter, I will start from a KPCA model of a set of phonological transcriptions, just as done in the previous chapters. These transcriptions lack syllable boundaries and stress marks, the placement of which is the *hidden feature* of each word. This placement is encoded by the fully specified transcription. So the full form is the set of hidden features, and the form bare of markup is the set of surface features. We can adopt this as a general view: There are only maximally specified entries in the data set, but depending on the task only a part of the entry is accessible. In the case of abstract features (chapter 3), the surface forms are easy to separate from the hidden features. Markers such as syllable boundaries are inserted into the word, so separating the layers is more difficult.

A kernel always is a certain *view* on the data. It models the distinction between surface and hidden features: All features implicit in the kernel are surface features, and we may choose to hide any features we want from the kernel. In the case of markup, I chose to represent this hiding directly: by having separate forms with and without markup. Then, the usual string kernel can work on the surface representations without having to explicitly ignore markup.[1] I will use n-gram kernels on

[1] Contiguous n-grams interrupted by markup are not represented contiguously, which would require a different kernel

both representations.

Remember that the representation returned for an unseen test item is fuzzy. It is a combination of training items, thus somewhere in between them. The place of the stress mark will thus not be unambiguous, but a distribution over possible places. Given that each training item has exactly one main stress, every linear combination will also have exactly one. The task is then to find the *optimal* place. Placing a single symbol into a string is the simplest form of markup. Just like identifying gender by majority vote (see 2.4.4 and appendix D), it is a form of *pre-imaging*: Finding a discrete value for a fuzzy representation.

The markup pre-imager is discussed in section 5.2. There will be no walkthrough for the methods of this chapter. The method of mapping unseen data items into feature space is identical to the one discussed earlier (3.1) String pre-imaging is introduced in 5.2 and demonstrated later, in the next chapter (6.3.3), because the examples in this chapter are too simplistic. This chapter presents results for stress annotation and syllabification in 5.3, and concludes in 5.4.

5.2 A pre-Imager for markup

Points in feature space are not interpretable results if the the original space is discrete. One needs to find the discrete value closest to the point in feature space. In the case of abstract features, this pre-imaging was done via exhaustive listing of all possible values (the three gender values). Each of them had a loading, and the one with highest loading was chosen.

Here the pre-images are strings themselves. The set of all strings over a finite alphabet is enumerable (e.g. by increasing length), but infinite. Upon encountering a close match there is no guarantee that it is in fact the closest possible. A more efficient strategy of searching the original space is called for.

Furthermore, the outcomes are not features, and therefore do not come with a loading, which could be directly used to compare the closeness to the point in feature space. Here, the kernel can help. The kernel value of an item and a linear combination of items is the weighted sum of the pairwise kernel values (appendix, A.1.4). Now one could maximize the kernel value *ad infinitum* by adding ever more stress marks and characters. A penalty for excessive material is given by normalizing the kernel (appendix A.1.3): Only a perfect match will have a kernel value of 1, all others will be below, sorted by proximity.

I employ an iterative search of the original space. The general algorithm for string pre-images is given in the appendix (D.3). It proceeds in minimal edits to the previously best approximation. As minimal edits I choose insertions, deletions and substitutions of single characters (as in Levenshtein distance). In the current application of markup, one in fact only needs insertions, if one starts not from an empty string but from the input item. The insertion can take place anywhere in the string,[2] so the number of items to check in each step linearly depends on both the number of insertable symbols

implementation. Conceptually they are equal.

[2]Further specialization of a pre-imaging method could impose linguistic restrictions here. For now, I keep the method as general as possible. The method is expected to sort out unattested placements.

and the length of the string.

There are good reasons not to start from the empty string. First, it takes many more steps to reach the optimal solution. Second, most characters of the output are already known. Each step bears the risk of running into a local maximum. A local maximum is such that no minimal edit will improve the kernel value, but at least one of these edits in combination with further edits would. Since markup only requires single, local edits, this is not really an issue here. I will discuss it in more detail in 6.2.1, where more general string pre-images are needed.

With the right choice of an initial item (the input itself) and minimal edits (insertion of a marker `'` or `.`, at any position in the string), the search space can be reduced significantly while at the same time avoiding local maxima. In stress placement, the correct solution can be found in just a single step; for syllabification, each step can introduce a syllable boundary in a correct place. Main stress placement exemplifies single symbol insertion in section 5.3.1, syllabification shows multiple symbol insertion in section 5.3.2. Section 5.3.3 explores different possibilities to do both, and section 5.4 discusses the findings.

5.3 Experiments

I compare different combinations of stress placement and syllabification tasks. The training set are 1000 randomly chosen polysyllabic words from the CELEX fraction described in 3.2.1 and appendix G.2: German non-compound nouns. Additional 100 words were chosen randomly as testing data. The average syllable count is 3.003 for polysyllabic words and 2.832 for all words.

Different n-gram kernels are compared. Unigrams are useless here, as they do not know anything about the order of characters and therefore cannot identify places within the string. The general idea behind using an n-gram kernel to identify insertion sites for syllable boundaries is that this task is usually solvable locally, as also witnessed by the approaches discussed in 5.3.2.1.

Syllabification is needed orthographically in typesetting (line breaks), and linguistically in splitting phonetic transcriptions into smaller parts. Syllables are a universal concept in human languages, and their perceived boundaries depend (deterministically) on a language's phonology. Sometimes, morphology may override phonology: Morpheme boundaries may be indicated by syllable boundaries. This is true for certain morphemes (with stems being more likely to exhibit this feature) in certain languages; others, such as French, re-syllabify coda consonants into the onset of the following word if it starts with a vowel. To my knowledge, syllable boundaries are never phonemic, that is, a different syllabification will never change the meaning of a word (unless this goes hand in hand with its morphological structure). Therefore, syllabification is deterministic.

Likewise, n-grams may identify the syllable carrying main stress. This is however a more global phenomenon, as it may depend on a syllable's distance to the word edge (depending on the language). While many languages exhibit clear patterns of main stress assignment (e.g. word-initial stress as in Finnish), others are more complicated (sensitive to syllable weight etc.). In contrast to syllable boundaries, stress may be phonemic at least in some languages. Although German has very few

minimal pairs only contrasting in stress (['teː.noːr], *tenor* vs. [teː.'noːr], *tenor (singing voice)*), stress is often assumed to be lexical at least in a class of exceptional cases (see e.g. Féry 1998).

In German, main stress correlates with syllable weight: stressed syllables are always heavy. This leads to tense, long vowels in stressed open (coda-less) syllables. In the CELEX data all vowels in open syllables are annotated as being long (see the example just given), so the assignment task is not trivial given the phonetic form.

Each task (syllabification and stress assignment) is described and its results are discussed in one of the following sections. The results are cumulated in table 5.1.

(5.1) Percentage correct markups:

ngram	2	3	4	5	6
stress	0.637	0.623	0.610	0.612	0.619
stress (with syllables)	0.658	0.673	0.653	0.638	0.637
syllabification	0.815	0.807	0.759	0.727	0.712
simultaneous	0.515	0.548	0.521	0.498	0.494
consecutive	0.553	0.547	0.516	0.494	0.489
iterative	0.589	0.518	0.513	0.501	0.491

5.3.1 Stress placement

Placing a single symbol into a word is the smallest change possible. The first application, main stress placement, does just that. Unseen test items shall inherit the position of the stress mark from words they are similar to on the surface, as measured by the kernel. They are mapped onto a point in feature space, that is, they are represented as a linear combination of words *with* stress marks such that the same combination of their unmarked counterparts best approximates the test item.

This linear combination will have many n-gram features with stress marks, in the vicinity of different characters, with different loadings. Placing a stress mark at any position within the word will match some of them. The pre-imaging method described above will find the placement which maximizes the similarity (the distance in the feature space) to the linear combination. It will however not stop there, as it is a general iterative method and does not enforce the uniqueness of main stress. An item with several stress marks will fall outside the training space (as captured by PCA), but its distance will be minimal. So in order to accommodate other features, more stress marks might be inserted. This is done until no improvement is possible.

Main stress is predicted correctly in 63.7% of the words with a 2-gram kernel, and 65.8% when syllable boundaries are present in the input. Performance is generally lower for kernels with larger features, with occasional exceptions for trigrams. A disadvantage of large features is their sparsity: Most larger n-grams including a stress mark have not been observed in the training data. Stress marks occurred only in 1000 context, once in each training item. Only small context windows are frequent enough to allow abstractions. Wherever stress is placed, it produces a lot of unseen n-grams, thus effectively increasing the distance. This problem is best tackled with more data. However doubling the training set did not improve performance. Many unseen contexts remain.

Stress placement improves slightly with syllable boundaries. They reduce the number of possible sites, although misplacements are still possible. `tr` is a good onset for a stressed syllable, but if it is actually `str`, then placing the stress mark before `t` is wrong. A syllable boundary (or word edge sentinel) before `str` favors the correct placement, because every word in the training set has the feature `.'` or `#'`, which would not be given by the incorrect placement.

Main stress is not a local effect, but all kernel features are local. Therefore, the method sometimes locally decides to stress a syllable irrespective of other stressed syllables. There is of course the generalization that every word has exactly one main stress, but placing a stress mark where it is called for outweighs violating that. Additional stress marks do appear where secondary stress fits (and reasonably so): `#'SpItsby:b@'rai#` (Spitzbüberei, *rascality*), `#'StUka'tu:r#` (Stuckatur, *plastering*). Thus the method tries to treat the problem locally.

In the CELEX annotation of German, all open syllables have long vowels, regardless of stress. Some of them are stressed, and some are not. Now vowel length is usually a good predictor of stress. This potential is totally blurred by the annotation. With a word like `#'i:zo:p#` (Ysop, *hyssop*), the method has to guess where to put stress, and it guessed wrong. Then again, stress and vowel length in German are only highly correlated, without clear evidence as to which is primary (lexical). Just as unclear is the status of unstressed full vowels; they do not contrast length. So an annotation with short unstressed vowels in open syllables would have been viable. However, this discussion is irrelevant to this thesis. It remains to be said that the model's performance could have been better on a different annotation.

5.3.2 Syllabification

The method only really features its iterative nature in the placement of several symbols. In syllabification, the method will first identify the most reliable place for a syllable boundary, and continue to identify others, until no improvement is possible. Syllabification is a strictly local task, as opposed to main stress placement. So the n-gram kernels are far better suited for it. With a feature width of maximally 2 the model achieves correct syllabifications for 81.5% of the training words.

A typical error is to accept a syllable end without realizing that the remainder is not a full syllable: `#i:.zo:.p#` and `#ky:.n.hait#` (Kühnheit, *daringness*) both receive a syllable boundary after a long vowel, which locally makes sense. When inflected (plural `#i:.zo:.p@#`, forms of the base adjective `#ky:n#` such as `#ky:.n@r#`), this is where a syllable boundary appears. Phonologically, the final consonant might be placed into an extrasyllabic position as the potential onset of a following syllable. In that sense, the assignment is reasonable. The model however fails to correctly learn the omission of a syllable end when the following material is rendered as a coda.

The converse of this situation with long vowels is found with short vowels. Short vowels appear before coda consonants, or geminates. In the CELEX annotation, geminates are represented by a single consonant after the syllable boundary. Nothing marks them as geminates but the fact that they are preceded by a short vowel and not followed by another consonant. This creates the situation that sometimes short vowels seem to appear in open syllables. The model learns this and places a syllable

boundary too much: `#O.k.tant#` (Oktant, *octant*). A similar thing happens with complex codas: `#zEn.f.t@#` (Sänfte, *sedan chair*). Here, both `n.f` and `f.t` are good syllable breaking points. They outweigh the non-syllable `.f.`.

5.3.2.1 Discussion

80% accuracy on syllabification is not a notable result at all, because I assumed syllable boundaries not to be phonemic. Optimality Theory (Prince and Smolensky 1993/2004) posits a universal, violable alignment constraint of different linguistic layers, to be instantiated with orientation (left edge vs. right edge) and affected layer. In the current case it aligns morphemes with syllables: For every left edge of a morpheme, there shall be a left edge of a syllable. While it goes too far to discuss here to what extent this constraint applies in German, it gives an explanation for phonetically odd, morphologically motivated syllabifications. Mostly this involves onsetless syllables, and the insertion of a glottal stop, which is neither written nor given in the phonemic annotation: "Halbinsel" (*peninsula*) is [halp.ʔɪn.zəl], not [hal.bɪn.zəl].

In this sense, syllable boundaries distinguish meaning. If the morphemic structure of a German word is known, syllabification is deterministic. Unfortunately there is no easy way to incorporate the morphological annotation of CELEX (see G.2) into the phonological annotation. This would reduce the rate of errors.

Daelemans and van den Bosch (1992) build a model for a closely related task: Hyphenation of Dutch words. Hyphenation is the orthographic reflection of syllabification, with only minor differences.[3] Daelemans and van den Bosch train an exemplar-based model on 19,000 Dutch words, which gets 98.3% of all hyphens right. Word accuracy then is a little lower, because even the incorrectly hyphenated words will have boundaries in some of the right places. They used 7-grams to compare words. Dutch behaves similar to German in the relevant respects. The main difference to the KPCA method then is in training size. PCA is mathematically more costly, which I why I usually train on only 2000 words.

Bouma (2002) starts out with a hand-coded set of hyphenation rules for Dutch (implemented as a finite state automaton) and achieves 94.5% right away. The model is further improved by learning exceptional patterns from 260,000 words from the Dutch portion of the CELEX database (Baayen et al. 1996; 90% of their data set of 290,000 words), then achieving 98.17%. As for German, one needs to know the morpheme structure in order to fully predict syllabification. Neither is given in the training data, so the rules basically learn morpheme boundaries, where otherwise the syllable boundary would have been misplaced. Bouma goes on to train his model on short n-gram patterns as used in TEX's syllabification (Liang 1983), which are extracted from huge data sets. Both models achieve hyphen accuracy above 99%.

Schmid et al. (2007) train a Hidden Markov Model (HMM) on the phonological annotations of 300,000 German words from CELEX. This is again syllabification, not hyphenation. Their "one-

[3] There are additional constraints on hyphenation concerning the fluency of writing: Do not strand single letters (even if they form a syllable of their own), do not split digraphs, and so on.

vowel-per-syllable constraint" enforces the correct number of syllables. Assignment is 99.85% correct using a context size of 5 preceding phonemes. This compares to using 6-grams (including the phoneme in question). The remaining errors are attributed to unrecognized morpheme boundaries, incorrect annotations, loanwords with rare phonemes, and very few real syllabification errors. This proves that syllabification is perfectly learnable from enough data.

An HMM needs smoothing, otherwise the model would return a zero probability whenever it encounters unseen n-grams. KPCA models do not 'crash' the same way; unseen features simply do not add to the overall similarity. One might consider smoothing in the kernel (adding at least something even if the feature is unknown), but I see no immediate need for this. Thus KPCA does not need an additional and arbitrary smoothing parameter, which I consider an advantage.

For better comparison with these methods, I ran another KPCA modeling on 2000 data items using a 4-gram kernel, this time also including monosyllabic words. The latter are not trivial, because the model might erroneously introduce syllable boundaries. I altered the pre-imager to reject syllables without nuclei, thus mimicking one half of the one-vowel-per-syllable constraint. The other half is to reject syllables with more than one nucleus (vowels separated by at least one consonant). The first constraint can be checked at each step of the iterative pre-imaging process; the latter cannot. Not implementing it seemed not to cause any additional errors.

The performance goes up to 95%, as compared to the previous 81.5%. Remaining errors are due to the very same reasons as listed by Schmid et al. and cited above. Taking the training set sizes (2000 vs. 300,000) into account, this compares quite favorably. The approaches discussed here outperform the KPCA model mainly due to their larger data set. In its current implementation, the KPCA method simply is not capable of handling such data masses.

There is however an objection to hard-wiring constraints into the pre-imager (or into the HMM): Such linguistic constraints should follow from the model. They should not be imposed on it. In the original experiment (5.1), I chose to keep the pre-imager as general as possible, so that any implicit knowledge about syllables would have to be learned. As the comparison to the model *with* the one-vowel-per-syllable constraint shows, this results in three times more errors. Eventually, with more data, the constraint should follow implicitly. However, even when training on the full CELEX database Schmid et al. relied on the constraint as a filter against errors.

5.3.3 Stress and syllables

I identified three possibilities to mark words for both stress and syllables. The first does it simultaneously, the second separates the phases and intermediately fixes a pre-image with syllables only, and the third performs the last approximation not on basis of the surface similarity, but on the similarity of syllabified forms. An upper bound for their performance is set by the the previous models' ability to separately predict the two types of annotation.

Simultaneous markup

Stress and syllable prediction can be done simultaneously. The set of insertable symbols comprises both the stress mark and the syllable boundary. The order of placement does not really matter; it depends on which gives the best increase in the kernel value. Everything that has been said about the two separate tasks applies here. Errors accumulate, and so the model achieves a mere 51.5%, again with the 2-gram kernel, as in stress placement. This is close to the hypothetical 51.9% predicted by multiplying the individual performances on stress and syllable boundary prediction.

Consecutive markup

Inserting stress marks and syllable boundaries as in the simultaneous markup can be split into two phases. First, a syllabified pre-image is found as in the syllabification task (5.3.2). In fact, in the modeling I use the very same outcomes as found there. These are then the initial items for the stress marking phase. I call this the *consecutive* method for multiple markup.

Unlike stress placement with syllabified forms (second experiment of 5.3.1), the initial item is not the correct one, but the prediction of the first phase. So all incorrect syllabifications will lead to incorrect final outcomes. On the remaining 81.5% correct intermediate outputs (see syllabification), it might perform as well as 65.8% (see stress assignment), but this was done with a KPCA on syllabified forms. This is done properly in the next section. Here, the linear combination is taken from the surface KPCA, and the syllabification phase does not add any information to it. Yet the correctly predicted 55.3% outperform even the maximally expected 53.6% (65.8% of 81.5%). This is better than the simultaneous method, and it shows that stress prediction on correctly syllabified words is easier than it is on average.

Iterative markup

The second phase of the consecutive method mixes input annotations: The linear combination was calculated by the Kernel KPCA on unmarked data, while the new initial item now already is syllabified. To properly benefit from the syllabified model of stress prediction, another KPCA is necessary. This model does not pass one linear combination through the phases of markup; it calculates a new approximation after each step of pre-imaging. This I call the *iterative* method.

One KPCA is performed on the unmarked training data, and another on the same data syllabified. The first phase is identical to the syllabification task, and thus to the first phase of the consecutive method. The syllabified pre-image is projected into the feature space of the second KPCA, and a pre-image with stress is found. This is identical to the second stress assignment experiment. This method achieves 58.9%, outperforming the two other combined prediction methods. As expected, this is worse than the two separate models.

5.4 Conclusion

This chapter provided a step from the prediction of abstract features to more concrete outputs, namely by adding information to strings. Following the main theme of the thesis, this is done via analogy: a test item receives its markup proportionally from the training items needed to represent that test item.

Pre-imaging methods were employed to convert the fuzzy feature space representation of a newly parsed item into a valid, discrete form of the space of annotated words. The pre-imagers were tailored towards the tasks in two ways: First, the initial item was chosen to minimize the number of steps necessary. Second, the minimal operations were restricted: only insertions of only the relevant symbols. This vastly reduces the search space, and makes string pre-imaging feasible. In chapter 6 I will use the very same pre-imager with less restrictions. More restrictions are possible, by hard-wiring linguistic knowledge into the pre-imager. However this defeats the generality of the method. In terms of human cognition this is the question of language-specific functionality. It is favorable to explain linguistic capabilities with more general-purpose mechanisms.

I also discussed several ways to predict multiple layers of annotation: simultaneously by giving the pre-imager all necessary symbols at the same time, consecutively by first finishing one annotation layer and then the other, or iteratively by performing a second KPCA on the intermediate layer. I chose syllable annotation as the primary layer and main stress marks as the secondary layer, following linguistic intuition. To extend this method to other tasks the linguistics, layers have to be pre-ordered by the researcher, or their most effective order could be determined experimentally.

In the example chosen, the overall precision was higher than the combined precision on the individual annotation layers, but only for the consecutive and the iterative method. From this I conclude that those can benefit from easy words: If it is easily syllabified, then stress assignment is more likely to be correct than on average. This explains how they outperform the predicted, combined precision. However, this seems not to be in effect with simultaneous annotation. Therefore I have to assume that the latter introduces additional errors due to the more complex pre-imaging process, bringing it back to the predicted level of performance.

The consecutive method only uses the KPCA model from the previous step. The further increase in performance seen with the iterative method then shows that each step should be grounded on a KPCA on the level of annotation in question.

n-grams are widely in use for similar prediction tasks. For example, all three approaches discussed in 5.3.2.1 use them. The kernelized method presented here allows maximal flexibility in the choice of surface features. I only used n-gram kernels here, but certain tasks might very well benefit from other surface features. The generalization over certain classes of phonemes is only implicit here, since different phonemes bear no similarity. A more sophisticated kernel could take into account the similarities of phonemes,[4] and thus more directly establish phoneme classes such as consonants and vowels. After all, the consonant-vowel skeleton of a word already determines stress

[4] I have done preliminary experiments with such a kernel. Some issues remained unresolved, so I excluded it from this thesis.

and syllables to a large extent.

Syllabification of German words essentially is a solved problem (see 5.3.2.1), given enough training data. The KPCA method is not feasible for such large data sets, because the decomposition of a 300,000-by-300,000 kernel matrix is beyond available computational power. Arguably, there is a lot of redundancy in the data set, so the eigenvalues of the principal components would quickly decline. Most of the models predictive power could presumably be captured with only a few thousand dimensions.

In conclusion, the KPCA model cannot yet compete computationally with well-established methods such as the ones discussed in 5.3.2.1. It is at an advantage when explicit feature listings are unfeasible and when more flexibility of features is called for.

5.4.1 Outlook

The initial results with relatively small training sets (2000 vs. 300,000) show that a lot can be learned with these simple models. Markup models can be built for any linguistic layer, and, as shown, also for any cascade of layers. For each layer, a specialized kernel and pre-imager is needed. Here I only used n-gram kernels; kernels on the consonant-vowel skeleton of words might be helpful in more efficiently generalizing over the data.

Another type of markup is transcription (see also Pirrelli and Yvon 1999): Mapping from the orthographic annotation layer to the phonological, and vice versa. Initial experiments show that the model benefits from more data. The results crucially depend on the pre-imager, which is more complex than the ones used here and in the next chapter (6.2.1): It needs to replace each orthographic symbol with a phonetic symbol (approximately). The number of steps is thus linear in the length of words. This opens up many possible dead ends (local maxima) for the pre-imager. I leave these results to future papers.

Chapter 6

Relating spaces: mappings and paradigms

The previous chapters have shown mappings onto abstract features and, less abstractly, onto enriched (annotated) versions of data items. This chapter now takes a further step and allows full flexibility in the co-domain: Both the the domain and the co-domain are the word space. Again, a mapping is to be learned based on similarity. As before, Kernel PCA models will be built on a training set in the domain, and unseen data items will be mapped into a feature space. The hidden features, however, will be inflectional forms. In order to pin them down, the pre-imaging process as presented in 5.2 needs to be made more flexible.

Learning by analogy does not only apply to learning abstract features. Learning paradigmatic relations via proportional analogy goes beyond learning as classification (Pirrelli and Yvon 1999). It extends to surface-level outputs. Words are linguistically related to other words, e.g. within an inflectional paradigm.

Paradigms for different words are analogous to each other, in that a paradigm for a certain part-of-speech always specifies the same slots (cf. de Saussure 1916). To simplify matters, consider only the relation between two forms in the paradigm: a so-called base form, and an inflected form. There is reasonable hope that this relation is systematic to at least some extent, when comparing different lemmas. Across languages one finds systematic rules, or several classes, with exceptions and sub-regularities. This calls for data-oriented models.

The paradigms I will be looking at are instantiations of inflectional morphology. The relation between rule-based and data-oriented models for inflectional morphology is discussed as the *past tense debate* (see 6.1), since irregular verbs in English are a classical case of study. This case is however largely dominated by a class of regular verbs, and sub-regularities within the exceptions are only instantiated by very few examples. There are far more interesting cases with *minority defaults*: The regular case does not coincide with the most frequent case. This is really a point in favor of data-oriented modeling, since rules have a very low coverage under such circumstances. In this chapter, I discuss English past tense as the classical example (6.5) and German nominal plural as an especially notorious case (6.4).

The accuracy of the KPCA models depends on having an appropriate pre-imager, since the space of all possible words needs to be searched systematically and goal-oriented. Such a pre-imager

is presented in 6.2.1, building on the one from last chapter (5.2). It proceeds iteratively, deleting or inserting symbols as necessary. Necessity is determined by the difference in symbols. This is calculated using a kernel on those symbols as features (that is, a simple unigram kernel), calculating the difference in loading for each feature between pre-image candidate items and the target (a point in feature space). The features have to be made explicit, but this is no concern with a unigram kernel.

The dimensions of the model describe surface variation, as usual, so they are not more interpretable than they were before (3.2.2.1). Note that I will be using the same data set for German plural prediction that I used in gender prediction (3.2). Since the variation in the output layer (plural forms) is about the same as in the input (singular forms), rotation (4.2) does not apply. See 6.4.5.1 for a discussion. To obtain a classification of the exponents of plural in German, I classify the changes from singular to plural. This can be done via a trick with the kernel, by performing a KPCA on singular-plural pairs (6.6), where the singular is counted as negative. The complete item will then be characterized by the list of feature differences. This is the same method as was just outlined for the pre-imager (unigram differences).

This *difference kernel* applies in general to all pairs of items which can be described by the same kernel, that is, by the same implicit features. To my knowledge, this is a novel application of kernels.

This chapter begins with an introduction to the task, learning inflectional morphology via proportional analogy (6.1). The KPCA feature inference method of the previous chapters is adapted to the task in 6.2, paying special attention to the pre-imaging method (6.2.1). A walkthrough is given for the German plural task (6.3), which is then executed in full (6.4). In addition, I look at the original task, English past tense (6.5). Section 6.6 uses the difference kernel to extract the classes of German plural formations. Section 6.7 concludes.

6.1 Paradigms and the Past Tense Debate

Traditionally seen, morphological processes follow rules, except — of course — for their exceptions. In modeling a language, one might simply list them, which seems to suffice for languages like English. However, already in the English past tense, this misses the regularities within the exceptions: irregular verbs tend to cluster into small groups with similar inflectional patterns (for an overview, see Keuleers and Sandra 2003). It certainly fails to explain *minority default inflections* such as Arabic (McCarthy and Prince 1990) or German plural formation, where the linguistically established regular case amounts for only 25% (Arabic) respectively 6% (German) of tokens in corpora (Plunkett and Nakisa 1997, Marcus et al. 1995). The discussion of improvements to the simplistic view is known as the *past tense debate* (see Pinker and Ullman 2002).

Proponents of rule-based (*dual-route*) models are e.g. Pinker and Prince (1998) and Marcus et al. (1995). Alternatives often come in the fashion of connectionism, which denies the strict separation into regular and exceptional cases and favors a single mechanism (*single-route*; Nakisa and Hahn 1996, Hahn and Nakisa 2000, Rumelhart and McClelland 1986, McClelland and Patterson 2002, among others; see Plunkett and Nakisa 1997, for Arabic). Rules only emerge as epiphenomena. In

this thesis, I follow a connectionist approach (see also the introduction, 1).

Connectionist models rely on similarity values (e.g., exemplar models) and/or surface feature representations (e.g. in networks, where each input node encodes a binary surface feature). At the current state of the art in machine learning it is only consequential to explore kernels for this task: Similarity and inference algorithm are strictly separated in kernel methods, and any measure of similarity can be used, without having to refer to explicit surface features. Kernel methods then are mathematically precise and do not have to be trained. In theory, this should yield computable and adequate models of inference via analogy.

In this thesis I explore only one kernel method (KPCA), and the question whether it holds up to this expectation. So far, a model for inferring abstract features has been built (3.2.2). In the current chapter, I extend the KPCA method for abstract feature prediction (2.4.3) to prediction of inflectional forms. The paradigms of unseen words can thus be modeled solely by analogy to existing words.

An interesting case to study is German plural formation, for its several competing classes, where the default class covers only a minority. A linguistic description of the data reveals a plethora of influences, mostly in terms of similarity to other words. With the general KPCA method, I will build a model of German plural, achieving up to 77% accuracy on unseen nouns.

6.2 Method

This section describes how to employ the KPCA method for inflection by analogy. In that sense KPCA instantiates connectionism. The task is to compute a plural form, given the singular form of a word. The main assumption behind an analogical model is that a singular form relates to other singular words just as its plural form does to other plural words. Using KPCA with a kernel over singular forms, the singular form is given as a linear combination of singular words. Now the plural form can be estimated as a linear combination of plural forms, namely the plural forms of the same words, with the same coefficients.

The coordinates of the singular form in the feature space of singular forms can be calculated from its kernel vector. The plural form is estimated as having the same coordinates in the feature space of plural forms. From this point, a pre-image is to be drawn.

The method generalizes in a straightforward manner. Whenever there is multi-level annotated training data, and only partially annotated testing data, one can infer the missing levels by assuming that the relation between data items is similar on all levels. Further examples are mentioned in section 8.2. All the linguistic knowledge, all what constitutes similarity in the researcher's eye, goes into the kernel — and kernels are already available for many levels of linguistic annotation (see 2.2.3.3). New and different kernels can be defined with ease.

The trickier problem is that usually in theoretical linguistics the data spaces are discrete, while linear combinations of the training items are not.[1] Each data type does not only need a kernel,

[1] One might imagine applications in phonetics, where there are continuous variables such as formant values and phoneme durations; and similar tasks.

but also a pre-imaging algorithm which makes the general pre-image problem (finding the best discrete approximation of a non-discrete representation) computable. Because of the computational complexity it pays off to carefully design the pre-imager. In section 5.2, I devised a special-purpose pre-imager for markup insertion. In the following section (6.2.1), I extend it to be applicable to inflectional tasks.

6.2.1 A pre-imager for inflection

Section 2.4.4 introduced the pre-imaging problem and a general-purpose pre-imager (also see appendix D.2). It proceeds stepwise and monotonously. An initial item can be chosen in order to reduce the estimated number of steps, before a solution is found. This choice is highly task-specific. In each step, the current best item is modified according to a set of minimal operations, each introducing none (if not applicable), one (if uniquely applicable) or several candidates. These candidates are close neighbors to the current item. They are evaluated via their distance to the target (see lemma 1 in appendix A) — the point in feature space that is to be approximated. Only the best one is pursued. As a result the algorithm is monotonous. If no candidate exceeds the current item, then this is the final output.

A crucial factor is the choice of the initial item. In finding a pre-image for a linear combination of strings, starting at the empty string is tedious. For the present purposes it is legitimate to make use of linguistic knowledge: the inflected form is close to the base form, since they (most often) share the stem. Choosing the base form as the initial item guarantees a short number of steps and (thereby) avoids local maxima elsewhere in search space. In inflectional morphology, the pre-imager will have to deal mostly with affixes, but also with some internal changes (umlaut, stem grades, vowel harmony etc.). These changes determine the necessary operations.

The pre-imager for inflection will have as minimal operations insertions and deletions of single characters. As opposed to the markup pre-imager, these characters are not only the markup symbols, but the full alphabet (or the set of phonetic symbols, depending on the annotation). As with the markup pre-imager, this one will employ unigram differences to prune the search space. These are extracted explicitly from the current candidate in relation to the target. Only those symbols are deleted/inserted for which the count in the current item and the target differs significantly (say, above 0.5 or below -0.5).

Insertions and deletions together yield substitutions. The intermediate form (after applying one of them) is somewhat unnatural and might lead to a premature abortion of pre-imaging with an incomplete output, because crossing it violates monotonicity. See 6.3.4 for an example and discussion.

To sum up: The pre-imager for inflection starts at the base form, looks which unigrams are missing or excessive, and inserts or deletes them accordingly, such that with each step the current candidate comes closer to the target, until there is no more improvement. This pre-imager will be used throughout this chapter.

6.3 Walkthrough

This section uses the same data as the previous walkthroughs, and extends the feature inferring method (3.1) to inflection as described in 6.2. I repeat the data set in 6.1.

6.3.1 Data

(6.1) Training and testing data (repeated from 3.1):

singular orth.	plural orth.	sg. phon.	pl. phon	gender
#blau#	#blaus#	blaʊ	blaʊs	n
#traum#	#träume#	traʊm	trɔʏmə	m
#tau#	#taue#	taʊ	taʊə	n
#baum#	#bäume#	baʊm	bɔʏmə	m
#raum#	#räume#	raʊm	rɔʏmə	m
#sau#	#säue#	saʊ	sɔʏə	f
#frau#	#frauen#	fraʊ	fraʊən	f
#pfau#	#pfauen#	pfaʊ	pfaʊən	m
#stau#	#staus#	staʊ	staʊs	m
#saum#	#säume#	saʊm	sɔʏmə	m

Of the plural exponents — which are to be discussed in section 6.4.1 — four are witnessed in the data: Three morphemes (-e, -en, -s) and umlaut. Earlier I needed all three genders represented, now the data set manages to provide a broad variety of plural inflection. This is not necessary for the current purpose, but nice to have.

6.3.2 Mapping the data into feature space

A bigram kernel PCA is performed on the training data, identical to the one in the previous walkthrough using the same data set. See section 3.1 for the details. The principal components are of course identical to previous run. PCA guarantees that.

The feature space image of the test item #saum# is the same as before, only that now the hidden feature is the plural form, not gender.

(6.2) Linear combination of training data approximating #saum# (repeated from 3.10, now with plurals)

$0.865\times$ sau, säue
$0.334\times$ baum, bäume
$0.239\times$ traum, träume
$0.202\times$ raum, räume
$0.071\times$ stau, staus
$-0.006\times$ pfau, pfauen
$-0.185\times$ blau, blaus

$-0.227\times$ frau, frauen
$-0.294\times$ tau, taue

This image is the point in feature space closest to saum, written as a linear combination of training items. Of course this closeness (0.965 in terms of the normalized kernel value between original and image) could only be evaluated on the surface, that is, on the singular form. The assumption now is that the plural of saum is similar to the plural forms of the words it is similar to. Just as gender in section 3.1, the hidden features of the test item are inherited from the linear combination. In this case, the hidden features are the plural forms. Now there is no immediate way to see which word is described by 6.2. This is the pre-image problem. I now go through the specialized solution suggested for inflection and similar string-altering processes.

Multiplication and addition of word forms is undefined. In order to decompose them we look at them via a kernel. The surface kernel lends itself also for this layer of annotation. Its implicit features (unigrams and bigrams) can be multiplied and added. This results in the feature listing in 6.3. Features with an absolute value below 0.5 are not shown.

(6.3) Implicit hidden features of 6.2 (by the bigram kernel):

$-0.6427\times$ au
$-0.6427\times$ a
$0.7769\times$ me
$0.7769\times$ m
$0.7769\times$ um
$0.8227\times$ s
$0.8657\times$ sä
$0.9371\times$ #s
$1.000\times$ u
$1.114\times$ e
$1.348\times$ e#
$1.642\times$ äu
$1.642\times$ ä
$2.000\times$ #

6.3.3 Pre-imaging

The pre-image problem still is apparent: This is supposed to describe a German plural noun to the best of the models abilities, yet there are no words with fractional or negative features. Intuitively, one might read the description as follows.

- All relevant features are contained with a coefficient near 1. Other positive contributions to the word fall under the threshold of 0.5.
- The negative features clearly set the plural apart from a form without umlaut. The extreme values are due to the small model. A larger model would predict values around 0 if it was sure

about umlaut; and small positive values if it reserves some probability mass for a non-umlaut plural.

- Conversely, the umlaut features are over-represented (larger than 1).
- Manually, the output can be reconstructed by plugging bigrams with high positive loadings together. In the example given, `#s,sä,äu,um,me,e#` lead to a unique solution, even the correct one: `#säume#`. In the general case, pre-imaging will not be that easy. Also, ad-hoc estimation is not a general purpose method.

The general pre-image problem applied to this case translates as finding a word (a sequence of letters from the alphabet, edge-marked with the sentinels) as close as possible to this fuzzy description. In practice, a search algorithm is needed to identify that word, because the space of possible words is infinite.

The specialized pre-imager starts from the input (a reasonable trick for inflection) and proceeds in single steps of insertions and deletions. Which symbols are to be deleted follows from the feature differences between the input and the estimated plural. Unigrams suffice, since the pre-imager inserts or deletes only single symbols. The unigram differences are given in 6.4, also with a threshold of 0.5.

(6.4) Unigram differences between image and initial item:

$1.1144\times$ `e`
$-1.6427\times$ `a`
$1.6427\times$ `ä`

The unigram differences here induce three pending operations: deletion of `a`, and insertion of `ä` and `e`. This is exactly what is needed. The order of these operations is not important; the pre-imager will find one which maximizes the initial gains (best operation first). The sites of the operations do matter. The deletion has a fixed site, since there is only one `a` in the word. For the insertions, the pre-imager will try all possibilities and pick the best ones. it then takes the following steps:

(6.5) Trace of the pre-imager:

Current item	distance	operation
`#saum#`	4.303	initial item
`#saäum#`	3.558	insert `ä`
`#säum#`	2.765	delete `a`
`#säume#`	1.778	insert `e`

- Per definition, the distance decreases with each step.
- `#saum#` is the initial item, with low fit.

- Out of the three pending operations, the insertion of ä is the most fruitful. It adds the loading of 1.6427 twice to the kernel value between the candidate and the target, once for the unigram and once for the bigram äu. Also, it introduces the unseen (and unwanted) bigram aä, an unavoidable side-effect of the minimal operations. This is repaired in the next step.
- A deletion adds to the overall score in two ways: First, it removes an unwanted unigram feature difference. Second, this deletion creates the new bigram sä by removing the blocking a.
- The insertion of e finishes the pre-imaging, even at the right result. There is no minimal operation that could improve the output, so we have arrived at least at a local maximum. Although there is no proof, it is very likely that it in fact is global.

This finishes the demonstration of inflection via a KPCA model with the help of string pre-imaging.

6.3.4 Discussion

Finding a feature space image for a test item in order to infer its hidden features is just the same for inflection as it is for classification. KPCA models can handle both. The tricky thing about infinite, yet discrete co-domains (e.g. the set of plural nouns) is finding that discrete point, when all one has is a linear combination of training items. This walkthrough showed how a pre-imager specialized to small changes to strings solves the problem for German plural inflection. The following section (6.4) evaluates this finding on a larger scale. I conclude this walkthrough with a discussion of some properties of the pre-imager.

The intermediate candidate #saäum# has some disadvantages. Technically, the unseen features caused by the simultaneous presence of a and ä could make this step impossible: While the unigram difference feature calls for the insertion of ä, it might cause so many unseen larger n-grams that no improvement is possible. The same might then be true for first deleting a, which also introduces undesired n-grams. Together, the two operations improve the pre-image; it cannot be guaranteed that they will always to so even when applied consequently.

Linguistically, the process would be described as umlauting the vowel, and definitely not as separate deletion and insertion. One could allow a search depth of more than one minimal operation. This however defeats the purpose of minimal operations, and it allows totally unrelated operations to be checked simultaneously for their combined benefit. A more well-behaved extension of the pre-imager includes a *replace* operation. This is useful in a general purpose string pre-imager, and I have used it in previous versions. It enlarges the immediate search space (the number of candidates reachable with one operation), but it does so moderately.

In combination with the operation selection by unigram difference, one would have to decide which operations go in pairs and which do not. This is a linguistic decision, and I wanted to keep the pre-imager theory-neutral. There are ways to proceed from here, creating special purpose pre-imagers; even learning a set of admissible operations from the data set. These include learning methods other than PCA, and therefore do not belong into this thesis.

The pre-imager works well without a replace operation on the German plural data. Therefore I chose to keep the simpler pre-imager with only insertion and deletion.

There are yet more particularities of the pre-imager which deserve some discussion. It is possible to remove pending operations after they are applied, so they are not applied again. If that is needed, one might want to adjust their loadings: add 1 to pending deletions, delete 1 from pending insertions. Thus the pre-imager could apply them again, until they either fall below a threshold or do not lead to an improvement any more. The version here uses the latter method, the brute-force option. This again is a point for fine-tuning a pre-imager.

With pre-imagers there is always a trade-off between generality and specialization. Specialized kernels might run into local maxima, but when a solution can usually be found in a small number of well-restricted steps (as is mostly the case for inflection), they are to be preferred. They can be enriched with additional linguistic knowledge to prune the search space. General purpose pre-imagers are usually much slower because of a vastly larger search space.

6.4 German plural

This experiment takes the previous walkthrough (6.3) to the full data set. Section 6.4.1 explains the linguistic complexity of nominal plural in German, and why similarity-based models are appropriate. Particularities of the data are discussed in 6.4.3. The experiments (6.4.4) include a reduction of dimension (6.4.5) as in 4.1 and the plural prediction task (6.4.6). A rotation as in 4.4 does not apply (6.4.5.1).

6.4.1 German plural morphology

The German nominal plural has received some attention within the past tense debate (Marcus et al. 1995, Nakisa and Hahn 1996, Hahn and Nakisa 2000, and others). This is due to the existence of many classes, only one of which satisfies the criteria for being a default. In contrast to English plural or English past tense, it is not the majority class.

(6.6) Plural frequencies of non-compound nouns. Data from Hahn and Nakisa (2000), examples partially from Köpcke (1988).

			Examples		
Morpheme	Frequency	Percentage	masculine	feminine	neuter
+ən	2646	30.775	Bär(en)	Tür(en)	Alkali(en)
+n	1555	18.086	Bauer(n)	Schwester(n)	Auge(n)
+∅	1992	23.168	Adler	—	Fenster
+∅ +umlaut	35	0.407	Bruder, Brüder	Tochter, Töchter	Kloster, Klöster
+ə	1178	13.701	Fisch(e)	Kenntnis(se)	Jahr(e)
+ə +umlaut	239	2.780	Sohn, Söhne	Kuh, Kühe	Floß, Flöße
+s	571	6.641	Park(s)	Mutti(s)	Auto(s)
+ər	36	0.419	Geist(er)	—	Kind(er)
+ər +umlaut	54	0.628	Wald, Wälder	—	Volk, Völker
ʊm→ən	95	1.105	—	—	Datum, Daten
a→ən	81	0.942	—	Pizza, Pizzen	—
ʊs→ən	69	0.803	Radius, Radien	—	Virus, Viren
ʊm→a	40	0.465	—	—	Signum, Signa
+iən	6	0.070	—	—	Privileg(ien)

Table 6.6 shows the distribution of different plural morphemes in German, as presented in Hahn and Nakisa (2000) for the non-compound nouns in the CELEX2 database (Baayen et al. 1996). Their method of identifying non-compounds slightly differs from mine, yet this should not distort the main picture. Some of the morphemes are given as substitutions, denoted by →.

+s is the default plural morpheme, or rather a "last resort" (see Marcus et al. 1995). It applies to loan words, truncations, abbreviations, zero conversions and so on. It preserves the phonology of the word it attaches to, while the other morphemes may introduce syllable boundaries, undo final devoicing, or even alter the stem (umlaut). Thus one might say that +s more analytical than the other morphemes. It is faithful to the stem at the cost of violating well-formedness: That a German plural form shall end in a Schwa syllable, which is true for most other forms. This condition crucially restricts the applicability of the null morpheme (to stems which already end in a Schwa syllable) and governs the allomorphic distribution of +n/+ən. There are other restrictions of the application of +n/+ə/+er/+∅, which I will turn to in a moment. The crucial consequence here is that neither of them qualifies as a default.

There are hardly any banning conditions on +s. Yet it only applies to 6.6% of the word types in CELEX (and similar figures have been found for tokens in texts; see Marcus et al. 1995). So how is the application of the alleged default plural morpheme restricted? For one, by the lexicon: Existing forms block the rule. However, traditional rule-and-exception morphology would have to list 93% of the German nominal lexicon as exceptions. This is clearly a case against that view. Second, by the well-formedness condition: Words, which for lack of an accepted plural had to take +s, are slowly being adapted into the German language. The most prominent example is internal re-analysis of words into stem plus affix, such that the affix can be dropped in plural. ən then directly attaches to

the stem: see the a/ʊm/ʊs→ən-rule[2] re-analyzing Latinate nouns. This values the well-formedness condition above faithfulness.

Already from the Latinate plurals (original +a or adapted +ən) it can be seen that there are classes within the non-default plurals. I will call them *non-default* rather than *irregular*, because the latter term is misleading. Although they are not fully predictable, they are far from being arbitrary and depend on various factors. I will give a few here, some specifically for the German plural, some more general.

Gender

Feminine nouns never take +ər or +∅ (Köpcke 1988), and umlaut+∅ only in two instances ("Mütter", "Töchter"). Köpcke suggests that the latter is due to the definite article, which is the same for plural (all genders) and feminine singular, and thus with a null morpheme there would be no overt plural marking at all. Umlaut+ə is marginal with neuter words (ibid.).

Derivational suffixes

Suffixes (e.g. "-heit", "-ling", "-schaft") determine the inflection.

Phonetic restrictions

+ə cannot apply if the stem already ends in a Schwa syllable. +s cannot apply if the word already ends in /s/ ('s' /s/, 'z' /ts/, 'x' /ks/).

Phonetic similarity

The most established factor in connectionist models is direct (surface) similarity to other exemplars. Even dual-route advocates sometimes model sub-regularities within the exceptions by similarity. However, Albright and Hayes (2003) show that these effects are by no means restricted to non-defaults: Regular forms clusters, too.

Orthographic similarity

In a study on Dutch plurals, Keuleers et al. (2007) found that speakers are also influenced by similarly written words. This comes into play with heterographic homophones.

Semantic similarity

Ramscar (2001, 2002) showed that the elicited inflection of nonce verbs with phonological similarity to both regular and irregular verbs depended on perceived semantic similarity to those. However, this appears only in combination with phonology; there is no proof for a productive connection between meaning and regularity.

[2]Partly extended to o: "Konto, Kontos/Konten"

Köpcke (1988) further mentions animacy as a factor for the German plural.

6.4.2 Discussion

Although there are some clear-cut gaps due to phonological principles (e.g. the ban on subsequent Schwa syllables[3]), morphology (no +ər in feminine nouns), and the lexicon (suffixes), most explanatory factors only express tendencies in the guise of similarity to other words and not rules. As such, they could be dismissed from a pure competence model. Yet non-word studies (Köpcke 1988) show that native speakers make use of such knowledge when judging or producing novel words. From large sets of regularly inflected words over small regions of quasi-regular non-default classes down to single exceptions treated lexically, there is no convincing evidence for a boundary between the rule(s) and the exceptions.

This all calls for a connectionist model able to capture several factors simultaneously, preferably in a modular fashion. I will now turn to the available data, before I lay out such a model. However, I will only be looking at a single factor in the present paper: the singular form. This is meant to primarily capture surface similarity, because arguably the singular form is most telling in predicting the actual shape of the plural: They tend to be very similar, except for a little suffix.

6.4.3 The data

The data is the same as used earlier (3.2.1): From the CELEX database (Baayen et al. 1996), I compiled a list of 13872 German non-compound nouns, of which there were both a (nominative) singular and a (nominative) plural form, in orthography and phonetic transcription. My particular implementation of compounds identified only and all noun-noun compounds by a simple heuristic: Two uppercase letters in the morphological analysis. All forms were enriched by an explicit word edge marker, #. This was to facilitate recognition of processes happening at word boundaries. Finally, all orthographic forms are converted to lower case. For more details, refer to appendix G.2.

CELEX has its downsides. It is somewhat prescriptive in its plural forms, often taking foreign plurals for loan words, along with the original pronunciation. This extends the set of phonemes to at least all French and English vowels. The transcription is phonemic, paying attention to certain rules (final devoicing, spirantization of /g/ after /i/), but not to all (no distinction between palatal and dorsal fricatives, orthographically 'ch'). So it is neither an underlying nor a surface form.

Further exploitable features are three classes of proper nouns, gender, syllable structure, morphological structure, stress, and more. The task at hand is the prediction of the plural form (both orthographically and phonetically), and any of these factors might contribute to the model. However, I did not include them yet. Similar points are being made by Hahn and Nakisa (2000).

Furthermore, I do not presuppose a fixed set of inflectional classes. This would give rise to a learning problem from singular form to abstract class, e.g. as in Hahn and Nakisa (2000), or as in

[3]This ban is not undominated in derivation ("Zauberer", *sorcerer*) and adjectival inflection ("sichererer", *a safer one*).

gender prediction (3.2). In line with connectionism such classes only exist as an abstraction of the data, and they would have to follow from the model. I believe the production of actual plural forms to be the more challenging and more insightful task.

6.4.4 Experiments

This section presents some experiments using the proposed method. The first experiment builds an initial model of the data and examines recall of words under restriction of the available information, thus condensing the model. The second experiment compares orthography to phonetic transcription. All experiments consist of a training phase and a testing phase. Training equals performing the KPCA; testing means to project every test item into feature space and calculate a pre-image.

6.4.5 Reducing the dimension

Section 4.1 showed that not all principal components are needed to faithfully remember the gender of each training item. Here, I test the same for plural forms. For this, I first build an exemplary model of German plural, using 2000 training items, 50 test items, and a 4-gram kernel on the orthographic annotation level. This is my usual exemplary model, only this time I reduced the test items by a factor of 2 to speed up the evaluation. 10 such models are built to obtain an average result. In the full prediction task (6.4.6) I will compare parameter settings; here it suffices to have just one model.

Information is spread unequally among the principal components. They are sorted by explanatory potential. A way to compress the result of the KPCA is to leave out components with low eigenvalue. A cut-off, as examined in this experiment, limits the available information not to the most informative single exemplars, but to principal components.

The model starts out with a KPCA on the training data. Each test item is mapped onto a point in feature space. This image cannot be interpreted directly as a plural word form, because the space of words is not continuous. The pre-imager for inflected word-forms (see 6.2.1 for the description, and 6.3 for an application in detail) now has to find a word as close as possible to the image in feature space. Only exact matches count.

With the same model, the evaluation is done by looking at all dimensions of the images in feature space and at restricted sets of dimensions, first discarding the lowest ones. Thus information is lost, leading to a decline in accuracy. The results are shown in figure 6.1.

The model starts out with around 80% correctly predicted forms. This value is not to be interpreted here; see the next experiment for a discussion. For now, it is just a point of reference: the model's performance under full information. The full rank of the models (number of dimensions) is between 1991 and 1997, with an average of 1994.3. Thus there is very little redundancy in 2000 data items as seen through a 4-gram kernel. At 1200 components, the model still reproduces as many forms correctly as the full model. As discussed in 4.1.1, this might also be due to the loss of information on too specific patterns, thus correctly applying a more general pattern, while at the same time some words start to produce erroneous forms. Performance with 400 components still is

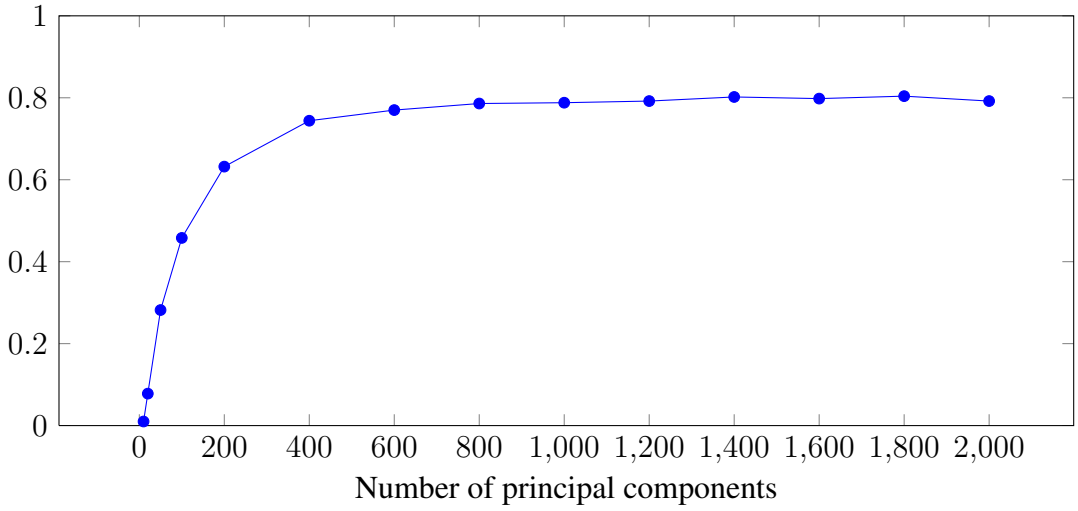


Figure 6.1: Plural prediction

at 74.4%, but after that the model drops towards zero.

It is hard to give a baseline. Random insertion and deletion of characters will almost never lead to a correct output. A better baseline is identity: Keeping the singular form as a plural form. Zero morphology applies to 18.2% of the words in my corpus.[4] The model drops below that, because with only a few principal components left it is not even able to correctly represent the singular form. It will thus distort the initial item, away from identity.

6.4.5.1 Rotation

For abstract features, a solution to retain all information relevant to the task was found in rotating the principal component basis, so that some contained this information, and the others were totally ignorant of the abstract features in question. This worked well for gender (4.4). However it cannot work directly for the current case, because the classes are not mutually exclusive abstract labels. Umlaut can occur together with some of the morphemes, +ən and +n are in allophonic distribution, and so on. The rotation as described in 4 applies only when classes have been pre-defined.

Furthermore, to specify the singular forms exactly, almost as many factors as training items are needed. The dimension reduction shows that about half of the principal components can be discarded safely. The plural forms now are as complex as the singular forms. To specify them, just as many principal components are needed. The order of principal components as given by KPCA is already optimized for representing singular forms; it will not be too far from optimal for plural forms, because they are very similar. Thus there is no room for improvement.

[4] Nakisa and Hahn (1996) report a type frequency of 16.840% in CELEX, and 23.168% for non-compound nouns. The difference is due to the extraction of non-compounds.

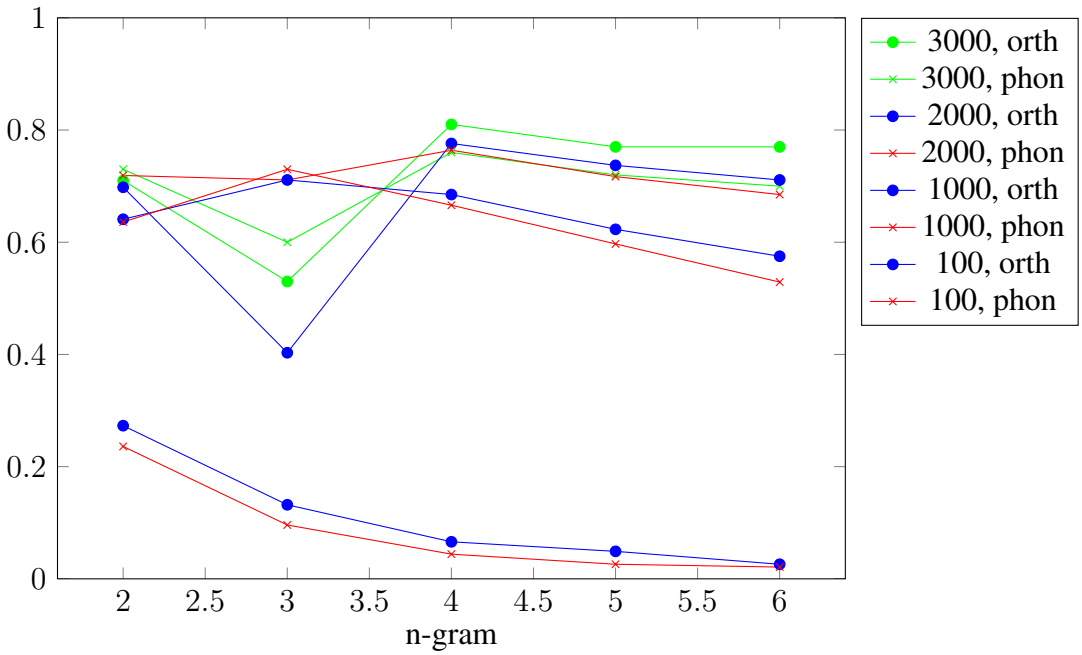


Figure 6.2: Plural prediction accuracy

6.4.6 Prediction

This is the full experiment on plural word form prediction. It compares different sizes of training data (10 as a baseline, 100, 1000, 2000, and 3000) and different kernels (n-gram kernels with n ranging from 2 to 6) on the two different annotations: orthography and phonology. The training sets are randomly chosen, one set for each size, to be used with each kernel. This maximizes comparability.

A KPCA is performed for each training set and kernel. Each test item is mapped into the feature space based on its kernel vector, and consequently mapped back into the space of words using the pre-imager (6.2.1). The pre-image is considered correct if it exactly matches the plural form given in CELEX.

(6.7) Training items: 100

ngram	2	3	4	5	6
orth	0.273	0.132	0.066	0.049	0.026
phon	0.236	0.096	0.044	0.026	0.021

Training items: 1000

ngram	2	3	4	5	6
orth	0.641	0.711	0.685	0.623	0.575
phon	0.636	0.730	0.666	0.597	0.529

Training items: 2000

ngram	2	3	4	5	6
orth	0.698	0.403	0.776	0.737	0.711
phon	0.719	0.711	0.764	0.717	0.685

Training items: 3000 (1 run only)

ngram	2	3	4	5	6
orth	0.71	0.53	0.81	0.77	0.77
phon	0.73	0.60	0.76	0.72	0.70

For each kernel and training set, the average rate (over ten runs) of correctly produced outputs is shown. Only for 3000 training items, I ran a single experiment with each kernel.

As the task was not to simply choose a morpheme out of a given set, there are more error classes than just this. A few are due to local maxima in pre-imaging. In others, rare or unseen n-grams were avoided at the cost of distorting the word. For example, `wunsch` might receive a plural `wuünsche` as a compromise between `u` and `ü` for lack of an n-gram `wü`.

Most erroneous forms are actually quite close to the intended or at least to a possible form. Possible orthographic forms are often tricked by one of the many loan words in CELEX: If read as a German word, the plural makes sense: "Korps" produced "Korpse" (correctly to be pronounced koːr, not kɔrp). For most errors, there is some sort of explanation. I expect results to improve on more data. However, the current implementation cannot cope with the full data set, in terms of computational effort.

The phonological models have special problems. Some possible forms are due to final devoicing: As the plural is not formed from an underlying form, but from the singular surface form, there is no way to tell whether a final obstruent should be voiced in the plural. Interestingly, there was both over- and underapplication of undoing final devoicing.

Bigrams sometimes are ambiguous: Whether the plural of `törin` (*fool (female)*) is rendered correctly as `törinnen` or incorrectly as `törinenn`, it has the same bigrams. This simple model has no means of distinguishing the two, and will consider both correct. This is also the reason why I do not use unigrams: The order of letters is totally random then.

6.4.6.1 Discussion

The model is able to fully recall seen data items. The performance on unseen data looks promising. The errors divide into real errors and possible forms. There is no automatic way to separate the two. Manual inspection showed 16% errors for orthography and 11% errors for phonology (at a run on 3000 training items, 4-gram kernel), the remainder being morphologically possible plural forms.

Possible means, the wrong suffix (and/or umlaut) was chosen, but had this been an unknown word to a native speaker, he could not clearly have rejected it.[5] Overgeneralizations included many exponents, even o→ən and +ər.

[5]Using the my judgment. In several cases even, the model and I agreed, in contrast to CELEX ("Verhalt, ??Verhalter/Verhalte", "Tribun, ??Tribunen/Tribune").

In comparison with the diversity of elicited plural forms in (Köpcke 1988, table 3), the model fares quite well concerning the correct and possible forms. However, native speakers do not run into local maxima. Their broader data basis enables them to accept more possible n-grams and words. This reduces their nonsense output to near-zero.

It can be said that the model actually employs a default: Identity. Starting with the singular form prevented many errors from happening at all, especially with null morphemes. This might be counted as avoidance of plural marking, which is found predominantly in early first language acquisition (Bittner and Köpcke 2001). Note that this is only avoidance of overt morphological marking, as these forms often appear with a quantifier.

The model cannot be reduced to a few inflectional classes, because at the same time it is a model of singular nouns in German. There is no separate mechanism preserving the stem (except for the choice of the initial item). It simultaneously models the choice of morpheme, allomorphy (+(ə)n), and double marking (umlaut + suffix). In fact, none of these are totally independent. Hence, abstract classes can never be separated from their instances, both in connectionist models and from a psycholinguistic perspective. In the present model, there is no clear cut (by manual inspection of the eigenvalues) between components modeling general tendencies (rules) and those responsible for so-called exceptions.

According to Marcus et al. (1995), the default plural arises especially when association fails (loan words, conversions, proper names). This could very well be a valid strategy for the language learner. Yet +s is not the only class which is overgeneralized in first language acquisition (Bittner and Köpcke 2001). Adult speakers have been exposed to all cases calling for a default plural often enough, so that a rule and an association-based model do not differ in their predictions. Association will however not be solely on a phonemic level; an according kernel would have to consider structural information as well. Consider as an example a zero conversion:

(6.8) $[_N$ Tiefe]+n (*depth(s)*) vs. $[_N\ [_A$ tief]]+s (*low pressure area(s)*)

In section 6.4.1 I have argued that +s is the most analytical among the plural suffixes. This is the basis for its wide applicability. A truly synthetic process such as umlaut cannot reach into internal structure: “Hochs/*Höhen” (*high pressure area(s)*). A kernel-based model could generalize the pattern $[_X[_Y \ldots]] \rightarrow [_X[_Y \ldots]]$+s, thus assigning +s to all conversions. Prediction then depends on the analysis given.

6.4.7 Comparison

Kovic et al. (2008) do a psycholinguistic study, with an underlying set of 5 rules:

1. -e for monosyllabic masculine nouns
2. -n for feminine nouns ending in -e
3. -er for all monosyllabic neuter nouns

4. -∅ for masculine and neuter nouns ending in -er/-el/-en

5. -s elsewhere

There is no percentage of coverage reported. The lack of any rules for umlaut makes this approach unfeasible. Although these 'rules' describe large sub-regularities, they are not without exceptions: "Mann, Männer" is an exception to rule 1, "Hemd, Hemden" and "Heft, Hefte" to rule 3. Rules 2 and 4 appear pretty stable, but rule 5, the default, comes far too early.

Mugdan (1977) is the classical rule-based approach to German nominal plural, with 15 rules/patterns and 21 classes of exceptions (cf. Köpcke 1988). So, more patterns can be extracted, and by this specialization towards surface features the account becomes data-oriented. Köpcke's study involves elicitation of plural forms for non-words. He constructed nouns of several types, including some with a clear cue (such as a derivational morpheme, or feminine nouns ending in -e) and other types for which a more diverse response is to be expected (cueless, monosyllabic nouns of any gender). Highest agreement (99%) is found with the morpheme -schaft (*-ship*), lowest with neuter monosyllabic nouns (40%). As agreement, I take the percentage of the most frequently elicited plural class, which would have to be considered 'correct' or 'grammatical'. Köpcke distinguishes only the five suffixes as classes here, since umlaut is very rare in novel productions. Weighted by items, the average agreement is 71.48%. Now this weighting surely does not reflect the real distribution of types; the precise reason for why there is so much disagreement with some types is that these are rare. Therefore I do not consider these 71.48% the expected inter-annotator agreement on German plural formation, I only take it as a lower bound and rather expect it to be around 80%, maybe even higher.

In comparison with other connectionist implementations (nearest neighbor: 71%, Nosofsky's Generalized Context Model (Nosofsky 1986): 74%, three-layer back-propagation network: 81%; all from Hahn and Nakisa 2000) my model is competitive (81% on 3000 training items with a 4-gram kernel), but those are predictions of the plural class (out of 15 abstract classes, see table 6.6), not of the actual output! All 'wrong' forms are 'possible' by definition. Hahn and Nakisa trained their models on 8598 data items, which even leaves some space for improvement to the present model. The items are represented by 240 features: 16 phoneme slots with 15 phonetic features each. The slots are aligned with respect to the right edge of the word, which means that by affixation the representation of the plural form is shifted to the left. This means the model cannot even pass the stem through unchanged. See Bullinaria (1997) for discussion of this 'alignment problem', which I hope is overcome by the use of kernels.

The KPCA model is a viable competitor to network models (Hahn and Nakisa 2000).

- It has the same performance (81%) on roughly 1/3 of the training data.
- It directly produces the output forms, instead of only classifying them.
- It only uses letters/phonetic symbols, so there is space for improvement by looking at the features directly, as done with the network.[6]

[6]However, in preliminary tests, a phonetic feature kernel produced errors such as /-z/ as a compromise between -s and -n, although word final /z/ is impossible due to final devoicing.

- It abstracts over the alignment problem.

Taatgen (2001) employs the ACT-R model, which separates into a declarative memory, where exemplars are stored for retrieval, and a procedural part, which provides an inflection analogical to the most similar exemplar in memory. He trains the model on 472 high-frequency nouns of German, and achieves around 90% in the end. The model has had a chance to see all items by then, so this is testing on the training set.

Daelemans (2002) trains a model on 25,168 nouns, with native plurals (first 9 lines of table 6.6) only. These are also taken from CELEX, so by the number I assume compounds are still included. Nouns are represented by hand-selected features: a CV-template of the phonemes of the last two syllables, and gender. Conflating the -ən and -n plural (predictable allophones), there are 8 output classes. Surprisingly, the model achieves 75% correct classifications with only 200 training items, and goes up as far as 95% with 20,000 items. The model is tested on the remaining 5168 nouns. van den Bosch (2002) reports similar results, but points to Daelemans (2002) for all details.

Other model (including mine) mis-classify four times as many unseen words, and native speakers can only agree up to a certain degree. The figure of 95% correct generalizations is meaningless unless one can show that human subjects are capable of such performance. Daelemans's model benefits from the lack of foreign plurals (about 3%, see table 6.6), from the presence of compounds, which means that unseen words actually have been seen, from gender information, from classification into 8 classes instead of predicting forms, maybe even from the hand-selected features being superior to general n-gram features, and from the larger training set. Still, I find it difficult to accept their results at face value. Outperforming native speakers (75% vs. 71.48%, see above) on only 200 training items cannot be possible.

The KPCA model fares quite well in comparison with others, with the one exception of Daelemans (2002). This is good enough for a proof of concept. At the same time, it models fully specified word forms, and therefore naturally needs more input data. This is rarely attempted (see 2.1.4), and in this respect the KPCA model is the best at hand.

6.5 Experiment: English past tense

Naturally, the English past tense is the classical example discussed in the English past tense debate. So it makes sense to apply the KPCA model to this problem. The data consists of 5866 English verbs (no auxiliaries) from the CELEX database. I also excluded complex verbs containing a space ("let in") or a hyphen ("window-shop"). For details, see appendix G.1. There are 311 irregular verbs in the data set.

Regularly inflected English verbs form both their simple past tense form and their past participle orthographically by adding -(e)d. A few things are to be learned: the alternation between -ed and simply -d, which is governed by the presence of an e; where to apply gemination of the final consonant (fit — fitted); the change from -y to -ied if it is not part of a diphthong (play — played vs. cry

— cried). Phonetically, the suffix is realized as [t] after voiceless consonants (missed [mɪst]), [ɪd] after voiced plosives (needed [niːdɪd]), and [d] elsewhere.

Irregular verbs mostly exhibit stem vowel mutations (see — saw). Also, the past tense form and the participle may differ. There are some, yet very small subregularities: a class of zero marking (put, let, cut), and patterns such as sing — sang — sung going back to regular classes of inflection by vowel mutation (ablaut) in the common ancestor of English and German.

The task consists not only in classifying a verb as regular or irregular, but also in applying all the necessary orthographic/phonological changes. A first experiment looks at regular verbs only. The second experiment then shows that from just a few irregular verbs nothing is to be generalized: unknown irregular verbs are always regularized. On the other hand, the model is able to learn any exception, since it can perfectly replicate the training set.

6.5.1 Experiments

Regular verbs

As usual, I used different n-gram kernels, both on the orthographic and phonemic layer, 2000 training items, and 100 test items. Average results from 10 runs are given below.

(6.9)

ngram	2	3	4	5	6
orth	0.927	0.920	0.943	0.932	0.929
phon	0.985	0.987	0.979	0.974	0.967

Typical errors of the orthographic pre-imager:

- Erroneous duplications: dine — dinned, void — voidded, regroup — regroupped
- Local maxima: remedy — remed. Here, only one step is required to reach an ending in -ed. Locally, this step is optimal.
- Unseen n-grams: lynch — lnched. It is better to leave out rare features.
- Double markings: backpedal — backedpedalled.

Errors of the phonological model:

- Local maxima: voɪd. This already ends in -ɪd.
- Unseen n-grams: frɒ (froth).
- Double markings: skeɪtɪdbɔːdɪd (skateboarded), blaɪndɪdfəʊldɪd (blindfolded).
- Wrong place: kɒndɪdɪsɛnd (condissended), kəʊɪdɪdɪnsaɪd (coincided), riːdɪdwɜːd (rewarded).

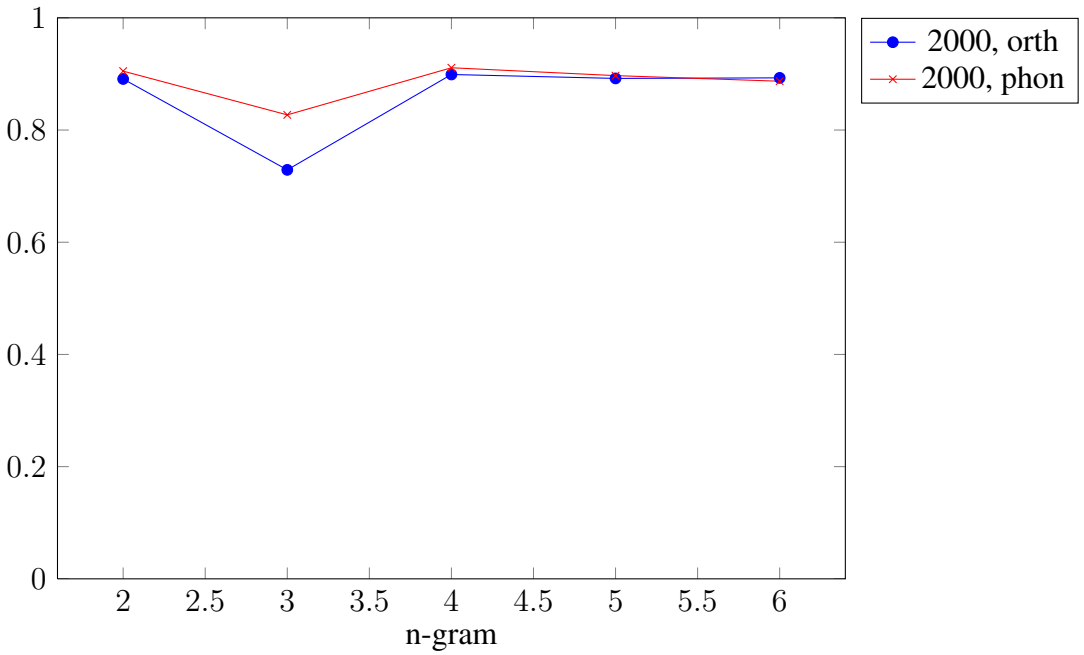


Figure 6.3: Past tense prediction accuracy

The method has an inherent problem with finding the correct place for morphemes, because of the position-independent n-gram features. This also leads to the double markings. For ‘skateboard’, the model will find a representation employing ‘skate’ and ‘board’ (ideally, that is, if these words appear in the training set; combinations of other words otherwise, just as these two would have been approximated). Both words would then throw their past tense exponents into the ring, and the pre-imager will make the best of it, which might not be correct.

Experiment including irregular verbs

The second experiment is just like first one, only that irregular verbs were allowed both in the training and the testing set.

(6.10)

ngram	2	3	4	5	6
orth	0.891	0.729	0.899	0.892	0.893
phon	0.905	0.827	0.911	0.897	0.887

Errors on orthography:

- Regularizations: catched, strived, selled, weared, forsweared. This is the main error source.
- Rare n-grams: pinnpointed, dissatisified, jinexed. These are ‘repaired’ by insertions.
- Erroneous duplications: reinned, steelled.

Errors on phonology:

- Local maxima: blaɪnd (blinded), riːd (redid), pəsweɪd (persuaded).
- Rare n-grams: dɒmətaɪzd (dogmatized), prɛstgæŋd (pressganged), leɪdɪsaɪzd (laicized/ˈleɪ.ɪ.saɪzd, with a very rare ɪɪ-sequence), bɜːdnd (burthen; due to a heavy influence from "burden"), and more.[7]
- Regularizations: swɛəd (sweared).

Different, randomly chosen data sets were used, so the results do not tell how the other representation would have dealt with the errors of the other.

Many of the forms being inflected regularly contain evidence towards irregularity to a small degree. I found cases where this surfaces. In the orthographic variant, 'bang' is so close to forms of the -ing/-ang/-ung sub-regularity that the initial item 'bang' itself already is a local maximum. In the phonemic variant, /stɪŋk/ realizes both the morpheme and a vowel change: /stʌŋkt/.[8] There are two possible interpretations of such a case. One is of model-inherent nature: The point in feature space identified as the past tense of /stɪŋk/ is actually a linear combination of /stɪŋkt/ and /stʌŋk/. Trying to approximate both with a single word results in the hybrid /stʌŋkt/. However, extending the search space to combinations of words is unfeasible unless one already knows the possible forms. Then one could extract coefficients for each form and interpret them as probabilities or as a measure of acceptance.

The second interpretation of the hybrid form is a linguistic one, where a grammatical function is actually marked twice. This is e.g. the case for the past tense of German auxiliaries: "konnte" (*could*) has a vowel mutation and the preterite suffix.[9]

6.5.2 Discussion

Orthography is more problematic than phonology because of the doubling of consonants, which is hard to learn for the model. Besides the linguistic difficulties, there are problems inherent to the method: Both models dislike unseen n-grams (a problem of strict data-orientation), run into some local maxima (especially when the initial item already looks roughly like a past tense form; a problem of the pre-imaging method), and exhibit misplacements and double-markings (a problem of the implicit feature representation by the kernel). These problems can be reduced by training on more data.

When also including irregular forms, these are largely outweighed by the regular verbs. Even a nonce word constructed to fall into the let/put/…class ('lut') is regularized more often than not.

On the larger data set, the advantage of phonemic annotation disappears, without an obvious explanation. Performance is around 90%. This compares to 80% reported by Plunkett and Juola (1999, fig. 6) on unseen verbs using a network. The input is encoded by 128 nodes, 16 binary phonemic features for each slot in a CCCVVCCC template. This general method is described in

[7] Curiously, all (and only) these words are also unknown to my automatic spell-checker.
[8] Plunkett and Juola (1999) also report automatic generation of forms such as 'tooked'.
[9] Historically, it is the vowel in *present tense* that is mutated.

more detail in 2.1.4. They model only monosyllabic verbs, so their data set is limited to 946 verbs. Therefore the models' performances are not directly comparable.

As any strict data-oriented method which does not classify, but gives full output forms, the template method also needs to learn all possible features of the stem and may encounter unseen n-grams. Also, the output feature vector may not describe an actual sound; Plunkett & Juola use a segment-by-segment pre-imager, finding for each feature bundle the closest English phoneme.

I consider 90% to be a rather low performance, and 80% even worse, but still the Plunkett and Juola model is the most relevant to compare the KPCA model to, since it maps from an input feature representation to an output feature representation, based on analogy.

The model by Taatgen and Anderson (2002) is geared towards children's learning of the English past tense: Simulating the U-shape in the inflection of irregular verbs. First, they are stored; then, the rule is learned and they are over-regularized; finally, evidence for the irregular form becomes stable again, and they are excluded from the general pattern. It is in this light that Taatgen and Anderson (2002) assess the literature on English past tense modeling, and that they test on seen data items only. Therefore the performance approaches 100% and cannot be compared to the present model. In contrast to simulating the U-shaped learning curve, the KPCA model describes a steady state, and does not need to be trained and updated.[10]

As in English past tense the irregular forms rarely surface, not the question of how to form the past tense is studied in the literature, but the question of how acceptable an irregular form would be. These studies are about predicting relative likelihoods / acceptance rates for either the regular or an irregular form, and thus goes beyond prediction of a single form. Accordingly, these model do not give a performance value directly comparable to the present model. Data is usually taken from CELEX (Baayen et al. 1996), and nonce items are classified into two (regular vs. irregular) or three classes (regular, vowel mutation, other) according to a similarity measure.

Eddington (2000) bases similarity on 10 hand selected features (phonemes in certain positions, and stress). Albright and Hayes (2003) criticize that this is tailored towards English past tense and not a general method for inflectional morphology. They admit a similar restriction for their rewrite rules in Albright and Hayes (2002): These cannot describe circumfixes (e.g. in German past participles).

In their own study, Albright and Hayes (2003) perform a production and a rating study, which both confirm *islands of reliability*: Subjects are more likely to produce irregular forms if the test (nonce) word is phonetically close to existing irregular verbs. Similarity is captured by edit distance with substitutions weighted for the similarity of the phonemes. Production is modeled by rewrite rules extracted from a data set.

Keuleers (2008) uses the Tilburg Memory Based Learner (Daelemans et al. 2007) to classify words by similarity and assign rewrite rules to new items. His rewrite rules are position-bound, in that he uses a CV-template for the final syllable only, aligning all items by its nucleus. By applying these rules, he generates fully specified outputs, not only class labels. In his model, islands are

[10]This would be possible by either adding more data to the training set and performing a new KPCA, or by using an incremental variant (Chin and Suter 2007).

characterized by taking into account only a small number of nearest neighbors. The more of them, the more weight is put onto regular inflection.

The most recent study along this line is Chandler (2010). In future work, the KPCA may be exploited in a way to obtain comparable results. According to Chandler, there is no established way to accommodate polysyllabic verbs, which were absent from the nonce test words. This is why Keuleers (2008) only encoded the last syllable. Kernels offer a way to overcome the problems of templates.

6.6 The difference kernel

This section introduces the *difference kernel* as a means to study the relation between two annotation layers. While an annotation cannot be directly subtracted from another, their feature lists can. Via the kernel trick these subtracted feature lists do not have to be made explicit. They can be compared (and, for instance, clustered) using the kernel. Section 6.6.1 defines this method. Section 6.6.2 applies it to German plural formation, thus identifying the possible patterns involved.

Note that a KPCA with regard to singular or plural forms (or even both at the same time) identifies variation in the word forms, not in the exponents of plural. Nowhere the model refers to plural formation. In order to learn something about inflectional classes, the primary data type in the factor analysis needs to be the change in each singular-plural pair. This change can be denoted by the difference in features. Imagine subtracting the singular from the plural, nullifying the stem and leaving only the suffix. Technically, data items are linear combinations of 1 times the plural form and -1 times the singular. From this follows the calculation (6.6.1 and appendix A.1).

6.6.1 Definition

The appendix defines how to calculate kernel values from pairs of linear combinations of data items. Assume that both layers are of the same data type (e.g., singular and plural forms are both strings), such that they are in the domain R of the kernel. Let x_1 be the first annotation layer of data item x, and x_2, y_1, y_2 accordingly. x is comprised of $1 \times x_1$ and $-1 \times x_2$. Then the difference kernel follows from definition 5 (cf. definition 10 in appendix A):

$$k_{diff}(x, y) := k(x_1, y_1) + k(x_2, y_2) - k(x_1, y_2) - k(x_2, y_1)$$

If one wants to classify the layer relations when the layers are of a different data type — with associated kernel k_1 and k_2 and their respective feature lists ϕ_1 and ϕ_2 — then the layer-internal kernel values are calculated using the appropriate kernel, and the cross-layer kernel values are 0 due to the non-overlap of features. The difference kernel reduces to a simple addition of layers. Thus it is only meaningful if the layers' features overlap at least partially. One then considers a new feature list ϕ_3 which contains the set union of the features of ϕ_1 and ϕ_2.

6.6.2 Experiment

I now apply the difference kernel to German plural formation. As seen from an n-gram difference kernel (plural minus singular), an item's positive features derive from the plural exponent; modulo some boundary effects, that is, larger features covering the suffix and part of the stem. Negative features are what is to be removed from the singular. These are mostly the features containing the final #, since what comes before is no longer word-final. Word-internal changes introduce more feature differences.

As an example, consider the pair "Baum — Bäume" (*tree — trees*) under a bigram difference kernel, encoded as #baum# and #bäume#. This example contains an internal change (umlaut) as well as a suffix. The unigrams and bigrams of the singular are #,b,a,u,m,#,#b,ba,au,um,m#, and those of the plural #,b,ä,u,m,e,#,#b,bä,äu,um,me,e#.

Their difference is $1\times$ ä,e,bä,äu,me,e#, $-1\times$ a,ba,au,m#. Note that these are quite a few changes. Over a larger data set, boundary features (covering the exponent and its surroundings) such as bä,-ba will be rare compared to the representation of umlaut ä,-a. One can therefore expect that boundary features will just be noise, whereas the real plural exponents emerge as stable patterns.

With a bigram kernel, I ran a KPCA on 1000 German nouns. In the following, I discuss the results. First, I turn to the data mean. On the negative side (features characteristic for singular), it contains — to various degrees — the following:

(6.11)
- The umlaut-able vowels a, u, o (in that order, which follows their relative frequencies)
- Many combinations of consonant + sentinel #

On the positive, plural side there are:

(6.12)
- The umlauts: ä, ü, ö.
- Many combinations of consonant + e (from either -e or -en).
- Most notably, the exponents of the most frequent plural classes: n, e, en, n#.

The data mean is no more than a statistical summary of German plural. The principal components show the dimensions of variation, that is, in which directions plural forms deviate from the mean. The first component is free from orthogonality constraints, as it is derived first. It divides the data set into the two most frequent plural classes: -e vs. -en.

(6.13) Predominant features of the first principal component:

$0.515\times$ n#
$0.507\times$ n
$0.491\times$ en
$-0.404\times$ e#

The second seems to separate the -e plural from -∅.

(6.14) Predominant features of the second principal component:

$0.587\times$ e#
$0.663\times$ e

Starting with the third component, arbitrary properties of the data outweigh the classes. This is a division of ...t → ...te and ...g → ...ge. Instances of these patterns would take a value of (roughly) +2 or -2 along this dimension. It is ranked so high because words in t or g (think derivations in '-ung') appear very often in the data.

(6.15) Predominant features of the third principal component:

$-0.503\times$ g#
$-0.469\times$ te
$0.477\times$ t#
$0.505\times$ ge

The fourth component is used with -s plurals, but it also plays a role in distinguishing the −t and −g cases from above. Such weird behavior is a result of forced orthogonality and the focus on remaining variance. Linguistically, it is not useful.

(6.16) Predominant features of the fourth principal component:

$-0.399\times$ t#
$-0.365\times$ ne
$-0.226\times$ g#
$0.227\times$ ge
$0.280\times$ n#
$0.308\times$ te
$0.371\times$ s#
$0.434\times$ s

Interpretability of the components further deteriorates from here. Therefore KPCA is not the right way to extract classes from the difference representations. In order to use rotation as described in 4.2.3, classes need to be known in advance.

6.6.3 Discussion

The difference kernel is a nice trick to get to the surface features in which singular and plural differ. Here I used a difference kernel on annotation layers of the same type: Both the singular and the plural are given as strings over the same alphabet. This allows for a short set of non-zero features for each item, when made explicit. Most features (n-grams of the stem) are shared and thus vanish

in the difference. Yet a difference kernel may be used to describe any sort of difference. Its feature set is the union of the feature sets of the kernels for each layer. In the current case, both layers have the same features, but this is not a prerequisite.

Feature differences are useful in determining edit operations, as seen in the string pre-imager (6.2.1). There, a unigram difference kernel extracts the necessary insertions and deletions to get from one form to another. Further, difference kernel representations of the data set could be clustered to obtain plural classes. However the dimensions of variation (the principal components) within the singular-plural differences do not tell much about classes, as seen in this section.

6.7 Conclusion

This method assumes that the singular and plural space have a shape similar enough that the mapping between them can be as simple as this: assuming that the plural forms is close to a linear combination of plural forms such that the linear combination of singular forms calculated to approximate the singular form has the same coefficients. As seen with the German plural, this assumption yields a linear model (as opposed to networks) which fares no worse than the best existing models (again, a network). In addition, it outputs inflected surface forms, not classes. This has rarely been done with similarity-based models. Notable are the network model used by Plunkett and Nakisa (1997) and Plunkett and Juola (1999), and the Wickelfeature model by Rumelhart and McClelland (1986). The relevant point is that the output layer structurally resembles the input layer, and does not only comprise nodes for different classes. A more thorough discussion is found in the introduction (2.1.4), here I only repeat and emphasize the points of the network model relevant to the KPCA models built in this chapter.

Plunkett and Nakisa build a network model for the Arabic plural, with explicit features both in the input *and* in the output layer. This goes beyond simple classification. The layers are given as CV-templates. Using a phoneme-by-phoneme pre-imager (finding the closest admissible phoneme to the weighting of features which is found in the output), they derive inflected output forms. Using templates allows for such local pre-imaging; with the position-free features of the n-gram kernel, I have to use a more complex pre-imager. On the other hand, the kernel is able to also treat words that do not fit into the template, and it does not suffer from the problem of how to align words with the template (see Bullinaria 1997).

The Arabic plural exhibits different classes, out of which two are regular (one for masculine, one for feminine nouns), but only used in cases similar to the German -s plural: derivations, conversions etc. The irregular classes are productive even for loan words, and hard to describe with rules. Plunkett and Nakisa's model achieves 95% correctness on the training data. Unfortunately, they test only classification, not production, on unseen nouns (in 10-fold cross-validation).

Their paper also stands out in that they use PCA to visualize the space of Arabic singular nouns. This PCA is performed on the explicit feature vectors that are also input to the network. The paper dates back to a time without Kernel PCA, otherwise they would have been well advised to use a

kernel. In conclusion, Plunkett and Nakisa (1997) is the closest predecessor to the present thesis: It maps from surface features to surface features in tasks of inflection rules are struggling with due to the analogical nature of the problem.

In case of competing evidence for different plural forms, the feature space representation of a test item will be a linear combination of plural forms with different exponents. Just as these points in feature space had loadings for each possible abstract feature (3.1), such a representation can also be interpreted as being loaded for different plural forms, as a weighted sum. Now hopefully the pre-imager will identify the most prominent of them, discarding the others. A more sophisticated method would recognize and savor the fact that several plural forms are superimposed here, and possibly extract all of them, along with their respective loadings. This conceptually more complicated interpretation might explain the relative proportions of different plural forms given by human subjects in production tasks. This is entirely parallel to the situation with gender assignment.

However, I do not see a general way to arrive at a superimposition of only a few possible plural forms. The search space for such representations is continuous, because even only two competing plural forms can stand in any proportion. A pre-imager for such outputs would have to be much more complex than the one for finding pre-images composed of single words.

6.7.1 Outlook

The method presented here extends to all paradigms in inflectional morphology. Depending on the task, the pre-imager might have to be adjusted, e.g. by allowing for substitutions. A paradigm extends to more than two layers, but it could be broken down to pairs of the base form and each inflected form. These would be learned as illustrated in this chapter, but the models for each inflectional form collapse into one, since there is only one model built (the space of base forms). The extension of the notion of analogical proportion to full paradigms is also found in Albright (2008) and Pirrelli and Yvon (1999).

I expect the method to deal well with affixation and internal changes (umlaut etc.), but I expect serious problems with other morphological processes such as reduplication. Although the kernel does not only look at presence or absence of features, but counts their occurrences, the representations cannot simply be multiplied by 2. The general pre-imaging strategy used here immediately runs into a problem when what is to be added is already there. Reduplication is hardly analogical on the surface. Rather it generalizes on some abstract level, where stems are available as a whole. The method is most useful in fusional/inflectional morphology.

Chapter 7

Applications to Syntax

7.1 Introduction

In this thesis I have touched three usages of KPCA models: classification, mapping to fully-specified forms, and interpretation of the extracted basis vectors (factors). All models so far concerned the space of *words*. This chapter explores possibilities in the space of *sentences*. Sentences can be seen as sequences of words and thus be modeled with string kernels. The other option is to use tree representations of syntactic structures. The main ingredient then is a *tree kernel* (Collins and Duffy 2001a,b, 2002, Shawe-Taylor and Cristianini 2004). Other structural representations (dependencies etc.) are possible, and appropriate kernels can be defined.

7.1.1 Classification tasks

There are some possible classifications for sentences. One might attempt to classify them according to their truth value (true or false), although it is questionable whether syntactic structure is informative with respect to truth.[1] A classification properly within the syntactic domain is grammaticality. Sentences are either grammatical or not, or one might assume graded grammaticality. Both categorical and graded judgments can be obtained from a KPCA model via the notion of a *planar language* (Clark et al. 2006): The data items define a subspace in feature space, which is the considered the space of all grammatical items. That is, linear combinations of grammatical items are in turn taken as grammatical. Graded judgments arise from the normalized kernel value between a test item and its projection onto the subspace, or from its distance to the subspace. Clark et al. did their study with string kernels; I take this to tree kernels in 7.2 (cf. the notion of *tree planar languages*, Costa Florêncio 2007).

Planar languages learn from positive data, and thus only have one class. Collins and Duffy (2002) train a kernel-based classifier on pairs of sentences with correct or incorrect parses, so they have two different classes. An external parser is used to list a set of parse candidates. This might be a simple Probabilistic Context-Free Grammar (PCFG). Candidates are then evaluated by the classifier,

[1] Except for the anecdote that "certainly" correlates with false statements, for which I currently lack the reference.

according to their distance to the separating hyperplane. For PCFGs, they use a tree kernel with subtrees as features (see the definition in A.3). These subtrees are defined as in Data-Oriented Parsing (DOP; Bod 1998). Collins and Duffy (2001a) also define a kernel for dependency structures, and one for paired sequences which is used in POS tagging. Basically, any type of syntactic structure can be assigned, when having a way of listing candidates.

Since their methods relies on negative data, and natural language is usually believed to be learned solely from positive examples (the language learner considers all input part of the language), I devise a similar candidate-based parsing strategy in section 7.2.6.

7.1.2 Mapping tasks

As argued earlier, while morphological processes are adequately described in terms of analogical proportion based on similarity, this is much less clear for syntax. Processes in question would be the mapping from sentences to sentence structures (with a tree pre-imager to be devised efficiently; cf. string pre-imagers in section 6.2.1), or to corresponding sentences in other languages. This can all be tried, but the simple linear model — estimating the output as the same linear combination of multi-layer data items that is needed to describe the input — is not likely to get us very far in this endeavor. I leave this to future research.[2]

7.1.3 Interpretation of the factors

The last usage of KPCA, interpreting the factors, has turned out to be infeasible earlier (4.6). I conclude that what cannot be done reasonably for string kernels will not yield meaningful results for syntactic structures either. Factors may be forced to denote certain surface or hidden features (see 4.5), but they will not uncover such features.

This chapter continues with my only application to syntax: applying the notion of tree planar languages to the artificial languages studied in Clark et al. (2006).

7.2 Tree planar languages

7.2.1 Introduction

In their 2006 study, Clark, Costa Florêncio, and Watkins show how even (mildly) context-sensitive languages can be learned using Kernel Principal Component Analysis (KPCA) with string kernels. From a training corpus, a model of the language is built which assigns goodness ratings to test items (see section 7.2.2). Some of the string languages they investigate have a natural representation as trees; others are not learnable by a string kernel. In this paper, I enrich these languages with tree structures, and apply a tree kernel (Collins and Duffy 2001b). The results largely depend on the

[2] In theoretical work, one is tempted to excessively use this phrase. I apologize.

information encoded in the trees, but even with simple structures the tree kernel always outperforms the string kernels.

Costa Florêncio (2007) takes this notion of *planar languages* to *tree planar languages*. Tree kernels encode information not contained in strings, such as grammatical relations. This is one step closer to parsing; however a tree kernel cannot be applied to a raw input string. Candidate trees have to be generated first. This can be done efficiently, e.g. with a context-free grammar (CFG) trained on the same corpus. The language model then identifies candidates with perfectly grammatical structure, or, more generally, ranks them according to their goodness rating. As we will see the tree kernel is able to capture restrictions beyond context-freeness. This gives rise to a parsing method: using a CFG to generate candidates, and the kernel-induced language model to sort them out.

7.2.2 Method

I adopt the method of Clark et al., with a few minor changes. In a nutshell, it proceeds as follows:

1. Input: a kernel, a set of training data, a set of test data.
2. Perform a Kernel Principal Component Analysis (KPCA) on the training data.
3. Generate grammatical and ungrammatical test items.
4. Project the test data into the hyperplane.
5. Evaluate:
   (a) Compute the distance (via the kernel, see lemma 1 in appendix A) between a test item and its projection.
   (b) If it stays below a threshold, classify it as grammatical.
   (c) Count the correctly and incorrectly classified items.

Whenever Clark et al. refer to strings (a kernel for strings, strings as training and testing data), I will use trees. Next, I will elaborate on each step.

7.2.2.1 Kernels

Classical PCA looks at vector representations of data items. By means of a kernel this representation never needs to be accessed explicitly. This is useful in handling complex data, such as natural language representations. One still needs to specify the features. Here, I compare three kernels: A tree kernel, and the two string kernels of Clark et al.:

- 1+2SUBSEQ: The subsequence kernel with features of length 1 and 2.
- GAPWEIGHTED: Likewise, with $\lambda = 0.5$.

For the definitions, see sections 2.2.3.1 and 2.2.3.2 as well as appendices A.2 and A.3.

In the tree kernel defined in Collins and Duffy (2001b), every subtree of a tree (in the fashion of data-oriented parsing; Bod (1998)) is a feature in its vector representation. A subtree is "any subgraph which includes more than one node, with the restriction that entire (not partial) rule productions must be included" (Collins and Duffy 2001a).

There always is one feature covering the whole data item, because the whole tree is a feature of itself. Therefore the kernel is injective (Costa Florêncio 2007): each data item is projected onto a distinct point in feature space. This makes generalizations (and computation, because of more features) harder, because unseen items will never exactly be recognized.

A variant of the kernel restricts subtrees to a height of h. A height of 1 yields the *bag-of-labels* kernel (cf. Costa Florêncio 2007). I will use $h = 2$ only: context-free productions. This kernel is strictly local, as opposed to the non-contiguous features in the string kernels. However, this is locality within the tree structure added to a string. The kernel can see long dependencies if the trees are constructed accordingly.

In the literature, there has been a slight confusion in terminology which may require clarification: In Costa Florêncio's (2007) more extensive collection of tree kernels this is mistakingly identified with Kashima and Koyanagi's (2002) *labeled ordered tree kernel*. They claim Collins and Duffy's kernel "is limited to trees where no node shares its label with any of its siblings". This is repeated by Costa Florêncio. The claim is true in Kashima and Koyanagi's understanding of a subtree, which allows including partial rule productions (contrary to Bod 1998, Collins and Duffy and Costa Florêncio!), because it is ambiguous which one of two identically-labeled sibling nodes is left out. In consequence, Costa Florêncio's definition of the labeled order tree kernel is identical to Collins and Duffy's tree kernel, not to Kashima and Koyanagi's. In Costa Florêncio's terminology, the kernel is best described as a *fixed height count-based all-subtree kernel*, but for the sake of simplicity I will refer to it as the *tree kernel*.

7.2.2.2 Normalization

All of the kernels used here tend to be peaked: they assign very high values to identical items, and comparably low values elsewhere. This can be overcome by normalizing the kernel function (appendix A.1.3). However, in the experiments to follow, this resulted in fewer correct classifications. When interested in strict grammaticality (i.e., exact membership in a language), a peaked kernel is actually a benefit.

7.2.3 Experiments

Experiments are conducted on several artificial languages described in section 7.2.3.1. For each of them, training and testing data are generated automatically and randomly. 100 training items and 500 (positive) testing items are drawn from the grammar, and an additional 500 (negative) testing items are constructed to be very close to grammatical items.

7.2.3.1 Languages

In order to investigate a string language by means of a tree kernel, an appropriate tree structure is needed. Trees encode additional information via their structure and via non-terminal node labels. Not all structures over a given grammatical surface string are grammatical themselves; typically, only one of them will be. Here is an overview of the languages (all taken from Clark et al.):

(7.1) Languages used by Clark et al.:

Language	Class	$\|\Sigma\|$	Definition	as context-free grammar
Even-1	REG	3	$\{\Sigma^{2n} \mid n \in \mathbb{N}\}$	$X \to X\Sigma \mid \Sigma\Sigma$
Even-2				$X \to X'\Sigma \mid \Sigma\Sigma, X' \to X\Sigma$
Even-3				$X \to XX \mid \Sigma\Sigma$
Bracket	CF	2	$\{ab, avb, vw \mid v, w \in \textbf{Bracket}\}$	$X \to XX \mid aXb \mid ab$
PalinDisj	CF	4+4	$\{vw^R \mid v \in \Sigma_1^+, w = z(v)\}$	$X \to \Sigma_1 (X) z(\Sigma_1)$
Palin	CF	4	$\{vv^R \mid v \in \Sigma+\}$	$X \to \Sigma_1 (X) \Sigma_1$
An–Dn	MCS	4	$\{a^n b^n c^n d^n \mid n > 0\}$	$X \to aXd \mid b (X) c$
Copy	MCS	8	$\{ww \mid w \in \Sigma^+\}$	$X \to \Sigma X \mid X\Sigma \mid \Sigma\Sigma$
GermScramb	MCS	4+4	$\{\pi(z(w))w \mid w \in \Sigma^+\}$	$X \to \Sigma X \mid X\Sigma \mid \Sigma\Sigma$

The Chomsky hierarchy class labels are REG for regular, CF for context-free, and MCS for mildly context-sensitive. z is a one-to-one mapping between symbols in Σ_1 and Σ_2. π denotes all permutations. **Copy** is Clark et al.'s **CrossDepND**. Some languages are altered to exclude empty strings.

Negative test items are drawn from closely related grammars, positive items from the true grammar. The context-free grammars (in most cases) merely illustrate the tree structures; they heavily overgeneralize. In these cases, negative items are drawn from a subset of the language defined by that CFG. This subset explicitly generates near-positives, which are especially hard to classify correctly. The CFG rules use a few shortcuts: | separates right-hand sides, parentheses denote optionality, Σ denotes any symbol in the alphabet, and Σ with an index denotes repetition of the same symbol.

Different strategies to construct trees are exemplified with **Even**. **Even-1** is trivial and uninformative, **Even-2** encodes information in non-terminal labels, and **Even-3** makes use of the tree structure. All other languages are described by CFGs with a single non-terminal label. So they all depend on the structure of the trees.

The three CF languages are naturally suited for a tree kernel. **Bracket** is the context-free language per se, the language of well-formed embeddings. **Palin** depicts center-embedding. **PalinDisj** is somewhat closer to natural language. It uses different alphabets, just as verbs and their arguments are not drawn from the same set.

Some of Clark et al.'s languages are defined entirely on surface counts: $a^n b^n c^n (d^n (e^n))$, $a^n b^m c^n d^m$ and **Mix**. These are suitably described using a string kernel, and the description lengths grow just linearly. However, the languages' complexity increases rapidly: a^n is a regular language, $a^n b^n$ is context-free, $a^n b^n c^n$ and $a^n b^n c^n d^n$ are mildly context-sensitive, $a^n b^n c^n d^n e^n$ is context-sensitive. Exemplary for the counting languages, I test $a^n b^n c^n d^n$.

The tree structures for **An-Dn** and **Copy** resemble the ones derived by schoolbook tree adjoining grammars for these languages. **Copy** has an upper, right-branching part, and a lower left-branching part. The scrambling language (**GermScramb**) resembles **PalinDisj** with free permutation in the first half. Its tree structure follows the one of **Copy**. For both languages though, it is non-informative: It does not establish direct connections between corresponding symbols, for this requires crossing branches, which in turn would require to specify a tree kernel for crossing branches.

Two context-sensitive languages from Clark et al. remain: **ChinNr** and **MultCop**. They exhibit dependencies which are covered neither by the string kernels' nor by the tree kernel's features. Therefore I do not expect the tree kernel to add any insights here.

7.2.3.2 Results

The presentation of results (table 7.2) follows Clark et al. FP is the ratio of false positives: negative test items wrongly labeled as positive; conversely, FN stands for false negatives. R is the rank, the number of dimensions in the data identified by the kernel. Values in parentheses are from Clark et al., for comparison; stated only when they differ.

(7.2) Results:

Language	1+2-SUBSEQ			GAPWEIGHTED			TREE		
	FP	FN	R	FP	FN	R	FP	FN	R
Even-1							100	0	11
Even-2	100	0	12	100	0	12	0	0	13
Even-3							0	0	9
Bracket	27.4 (10.8)	0	3	27.4 (10.8)	0	5	0	0	2
PalinDisj	0	0	20	0	0	30	0	0	7
Palin	15.4 (16.1)	0	14	17.8 (16.1)	0	14	0	0	7
An–Dn	0	0	24	0	0	38	0	0	8
Copy	100 (70.0)	0	72 (71)	100	0	72	0	5.0	67
GermScramb	0	0	26	0	0	51	0	0	19

Even has 3 symbols, hence 9 symbol pairs, hence 12 features, and hence a rank of 12. The first tree language for **Even** does not add any information, and thus the tree kernel performs as poorly as the string kernels. **Even-2** encodes evenness in alternating non-terminal symbols, **Even-3** does so solely via structure, using a single non-terminal. Both strategies let the tree kernel grasp the relevant information.

Not surprisingly, the tree kernel learns **Bracket**, **Palin** and **PalinDisj**. It does not treat the latter two differently (in terms of rank), which is an improvement over the string kernels. The difference in the reported FP value can be ascribed to the construction of negative testing examples. These values also differ from one run of the experiment to another.

Although its features are only (tree-)local — as opposed to the string kernels' unbounded features — the tree kernel can solve the counting problem **An–Dn** and **GermScramb**. **Copy** is most

interesting, as it is the only case with false negatives. The tree kernel clearly outperforms the string kernels here. **Copy** has 72 string features, and 80 height-2 tree features. The PCAs with string kernels have a rank of 72, and therefore could not abstract anything. The tree kernel PCA's rank indicates that the language has not been learned successfully, but it fares remarkably well for a MCS language. 100 data items are barely enough given 72 features, so I used 250 instead of 100 training examples. The tree kernel kept improving slowly on more data.

7.2.4 Towards natural language

The languages examined so far are highly artificial. This section looks at a small context-free grammar, which models a very small fragment of English down to pre-terminal nodes (lexical categories). The rules are as follows:

(7.3) S $\rightarrow$ NP VP | NP v
NP $\rightarrow$ d n | d N′ | NP PP
N′ $\rightarrow$ a N′ | a n
PP $\rightarrow$ p NP
VP $\rightarrow$ v NP | v NP PP | v NP S

In addition to the three kernels so far, tree kernels with features up to height 3 and with all subtrees are tested. The models are again built from 100 training items, but tested on 80 test items only. All items were restricted to a maximum height of 8. The main computational bottleneck is the parallel storage of all those items along with a pre-calculated list of all their subtrees, the number of which grows exponentially with tree size. Results are as follows:

(7.4)

1+2-Subseq			GapWeighted			Tree (2)			Tree (3)			Tree (all)		
FP	FN	R	FP	FN	R	FP	FN	R	FP	FN	R	FP	FN	R
12.5	0	18	38.7	0	20	0	0	6	0	0	22	0	37.5	48

String kernels are not able to learn context-free restrictions. The height-unbounded tree kernel is injective (Costa Florêncio 2007), and therefore cannot generalize to unseen trees. The threshold might be set differently, but it is generally unable to distinguish between a short ungrammatical item and a long grammatical item, as both introduce unseen material.

Given enough data, larger features (height 3) eventually cover the grammar. Yet the optimal solution is the one with the most appropriate feature selection: trees of height 2. In this experiment, results were the same for features with exactly height 3 and for features with up to height 3. So adding more features can only increase the rank, yet it does not if these features are already contained within larger ones (as are height-2 trees within height-3 trees).

A rank of 6 here indicates 6 dimensions of variation in the data. Each non-terminal introduces $n - 1$ dimensions, where n is the number of rewrite rules for this non-terminal. The remaining rule is reflected in the data mean. Consider the single *PP*-rule: It has no alternatives, and therefore

does not add any variation to the data. In the example grammar there are 11 rules and 5 different non-terminals; this leaves 6 rewrite rules as alternatives to an arbitrarily fixed default rule for each non-terminal.

The language model defines a hyperplane in feature space. For the height-2 tree kernel, every production rule is a feature. Given enough data randomly generated by a CFG, the KPCA establishes all features as independent (modulo the restriction that there has to be a rule with the unique start symbol of the grammar). Any *relative* proportion of features can be modeled. Now we need to see how this extends to all feature combinations in absolute terms.

Consider a test item larger than any seen training item. As the linear combination coefficients add up to 1, its subtree counts cannot be modeled as a linear combination of each training item's features. Clark et al. suggest to drop this restriction. This could also be achieved by evaluating test items via the *normalized* kernel (appendix A.1.3) instead of taking the distance. Then all items with a normalized kernel value of 1 (or close to 1) are considered to be inside the planar language.

Assume x can be modeled, and x' with $\phi(x') = a\phi(x)$ ($a > 0$) is to be modeled. This effectively means coefficients add up to any value a (instead of 1). Then the normalized kernel between x and x' is:

$$\hat{k}(x, x') = \frac{< \phi(x), a\phi(x) >}{\sqrt{< \phi(x), \phi(x) > \cdot < a\phi(x), a\phi(x) >}} = \frac{a\phi(x)^2}{\sqrt{a^2 \cdot \phi(x)^2 \cdot \phi(x)^2}} = 1$$

Thus items x' of any size are within the language as long as a reference item x with the same feature proportions is. Once the independence of the context-free productions has been established, every tree built via the CFG is accepted as grammatical.

7.2.5 Discussion

There is an upper bound to the rank given by the number of features. A high rank means little abstraction, a low rank means good abstraction. The tree kernel often succeeds to find a rank near the number of independent rules in the grammar, see **Bracket**. Even when the string kernels perform flawlessly, they often do so at a much higher rank, see **An–Dn**.

The most important lesson to be learned is that the recognition of a language depends on the tree structure superimposed. The regular language **Even** demonstrates this point. Informative nonterminal symbols or a clever tree structure both do the trick. In the former case, even the bag-of-labels kernel (Costa Florêncio 2007) will do.

The kernels induce language models on positive evidence only. A model *recognizes* members of the language, and approximates them with already seen members which have similar features. They crucially do not need to share any more structure than the one captured by the features. So this method does not *parse*.

The advantage of the tree representation is that it encodes grammatical relations. Consider **An–Dn**: `a`'s and `d`'s (and `b`'s and `c`'s) are organized in pairs. Although this language is mildly context-sensitive, the tree kernel correctly recognizes valid structures. Even for **Copy**, it gives a

good estimate of grammaticality. However, there is a way to exploit a tree kernel for parsing, as follows.

7.2.6 Parsing with tree kernels

A recipe for parsing, making use of a (tree) kernel, explained in more detail afterwards:

1. Input: A phrase-structure-annotated corpus, and a kernel
2. Read a CFG off the corpus.
3. Perform a KPCA on the same data.
4. Parsing: For each test string:
   (a) Generate parse candidates via the CFG.
   (b) Evaluate each candidate via the KPCA language model.
   (c) Rank the parses according to the distance (or the normalized kernel value) between them and their projection onto the hyperplane defining the language.
   (d) Pick the best one.
5. Optional: Update the language model.

Possible applications deal with natural language corpora, annotated with phrase structures. A CFG is read off a corpus by listing all occurring production rules. An input string can be parsed in cubic time using this grammar (and a standard, Earley-style chart parser), producing candidate parses (as in Collins and Duffy 2001a). Assumably the corpus exhibits some non-local dependencies, restrictions on distributional patterns and the like. As the CFG cannot capture them at all, it will largely over-generate. This is similar to the CFG descriptions I devised manually in section 7.2.3.1. As seen with the MCS artificial languages, the tree kernel can capture some restrictions, and thus find good parses among the candidates.

The method differs from Collins and Duffy (2001a) in the machine learning algorithm used (KPCA), and in that it does not need negative (ungrammatical) training items. The set of candidate parses for an input string is ranked by the KPCA language model trained on the same corpus, according to their distance to the point in feature space denoting the parse. The best one is chosen as the correct parse.

7.2.7 Conclusion

This section has shown how a simple tree kernel with local features outperforms string kernels in recognizing the languages studied in Clark et al. (2006). It is more precise, or it finds a solution of lower rank, or even both; also for context-sensitive languages. The crucial point is where to get the tree structures from. For the artificial languages, I have devised context-free grammars not relying

on informative node labels, but on structure alone. Either strategy works. For natural language, trees can be generated by a CFG read off an annotated corpus.

I have devised a method for parsing, employing a tree kernel to rank parse candidates output by a CFG. As suggested by the experiments on artificial languages, the kernel is able to capture some context-sensitive restrictions in the data. For context-free languages, the tree kernel with subtree height 2 finds the optimal solution.

7.3 Discussion

In this chapter, I have looked at classification as an application of (tree) kernels to syntax only. I have demonstrated the notion of *tree planar languages* (Costa Florêncio 2007) with the languages from Clark et al. (2006), as the logical next in this line of research. Grammaticality is defined as being close to the hyperplane in which all grammatical items lie. Dimension reduction can be used here for noise reduction; assuming that annotation errors and some ungrammatical items enter a corpus as noise. Collins and Duffy (2002) exploit kernels for parse re-ranking: sorting candidates by kernel value. It may also be possible to compare kernel values to graded judgments of grammaticality.

Parse re-ranking closely resembles pre-imaging, the difference being that they take a fixed candidate set as output by a parser. With a general-purpose tree pre-imager (imagine minimal operations on trees) and an efficient implementation parsing could be attempted directly with a kernel. Parsing would work just like all other applications in this thesis: Locate a sentence in the space of sentences according to its surface similarity to items in the training set, and find a pre-image for the same linear combination of the corresponding trees. According to C. Costa Florêncio (p.c., October 2009), A. Clark has tried to learn a mapping from sentences as described by string kernels to trees, but never published on the topic.

The syntactic structures do not have to be trees. Remember that the tree kernel takes all subtrees — as defined in Data-oriented Parsing (DOP; Bod 1998) — as features. There are DOP-approaches to Lexical Functional Grammar (Bod and Kaplan 2003) and Head-Driven Phrase Structure Grammar (Arnold and Linardaki 2007). Basically, any syntactic structure is some sort of graph and can thus be decomposed into partial structures, which can serve as implicit features for a kernel. Given an efficient implementation, the features do not have to be stored explicitly. Their large number is the major drawback of DOP. Kernels offer a solution to it.

The third use of kernels is also the most problematic: Interpreting the basis vectors in feature space (e.g. the principal components of KPCA) linguistically. I have argued earlier (2.3.4.3) that unrotated principal components are meaningless. A rotation towards features (VARIMAX) is in conflict with the use of kernels, however a rotation towards pre-defined classes is possible (see 4.2.3). The classes considered in this chapter were only grammatical vs. ungrammatical. As ungrammatical items are not within the hyperplane (by the assumption of planar languages), there is only one class, grammatical, into which all items fall. Just as the *three* gender values fell into a *two*-dimensional plane (see the gender experiment, 3.2.4), the problem of grammaticality (*one* class) has *zero* dimen-

sionality. In plain words: There is no vector to rotate. Therefore there is no interpretable factor denoting grammaticality.

Chapter 8

Conclusion

Following the goals outlined in 1.3, I will now conclude to what extent they have been reached with the models and methods presented in this thesis.

8.1 How to employ kernel methods in linguistics

The main application of kernels in linguistics is in classification, and the predominant kernel-based classifier is the Support Vector Machine (SVM; Cristianini and Shawe-Taylor 2000). Classification can also be done with Kernel Principal Component Analysis (KPCA; Schölkopf and Smola 2002, Shawe-Taylor and Cristianini 2004). In this thesis I have presented a way to get fully specified output forms instead of just classes, which is interesting in inflection (see chapter 6). The combination of analogical models and full output forms is rare; examples are the artificial neural networks of Plunkett and Nakisa (1997) and Plunkett and Juola (1999), and the Wickelfeatures of Rumelhart and McClelland (1986). Previously, linguistic tasks had to be broken down into binary classifications, see for example the semantic parser of Kate (2007).

8.1.1 Abstracting over exemplar models

Exemplar and prototype models differ from rules in that they place the knowledge about a class where the data is (that is, at the exemplars themselves, resp. at the center of the class), and not at its boundary, where exemplars are sparse and judgments largely differ. Graded class membership depends on closeness to the center (or prototype) in exemplar models, but on distance to the boundary in — for example — the parse re-ranking approach of Collins and Duffy (2002). In that respect a KPCA model is psychologically more valid than an SVM, because it classifies new items according to their similarity to others, and not via their relation to a decision boundary.

The model is built from individual exemplars, yet its final units, the principal components, never refer to single exemplars. Therefore it deserves to be called ‘connectionist’. New exemplars will change the model. Under real-world circumstances, every word’s many occurrences would be distinguishable by minimal (phonetic or circumstantial) deviations. These would be reflected in

low-ranked components. Leaving out those, the model gets more and more abstract.

Connectionist models rely on a notion of similarity between the data items. Kernels are the state-of-the-art technique in machine learning to define similarities. Kernels can be built for any data structure, according to the researcher's understanding of similarity in a particular case. Schölkopf and Smola (2002) devote an entire chapter (13) to the design of kernels.

8.1.2 Pre-Images

In order to interpret a point in feature space, a discrete pre-image has to be found. This can be a majority vote for mutually exclusive abstract class labels, but for concrete linguistic objects such as word it is much more difficult. In this thesis I have used special-purpose instances (e.g. in section 6.2.1) of a general pre-imaging strategy (D.2). The first trick is to limit the search space by choosing an initial item as close as deterministically possible to the output. In inflection tasks, this is the base form. The second trick is to define a set of minimal operations as small as possible. For string in general, these are insertions, deletions and substitutions of characters. The set of necessary operations can be further limited by evaluating the unigram differences between the current candidate and the target. Similar tricks would have to be devised for other data types and tasks.

An analogy to finding a pre-image is looking for something one only knows implicitly, something one only has a description of. This description is given as a combination of known data points: things you actually know. You would immediately recognize something that befits the description, but you could not create or make it explicitly. The question now is, what object in the real world best befits that description? Take the description of a chimera: part lioness, part goat, part snake. Once you have seen it[1], you know what it looks like.

Descriptions might be well-formed, but without an extension in the real world ('round squares'). In general, what you imagine (as a new combination of previous experiences) will not exist precisely in that form. A pre-imager will still try to find the next thing possible. It is not guaranteed that the human mind finds the closest match; it will be satisfied with the next best thing (i.e., it will run into local maxima). When presented with a better solution, it is able to adopt it. Brainstorming is a way of widening the search space, when the conventional solution does not feature all desired properties. There is a description of what the solution should look like, but the actual, possible solutions have to compromise. A measure of similarity (between the candidate and the mental description) will sort them out. In that sense, kernels capture associations, and pre-imaging relates imaginative things to real things.

8.1.3 Weighted KPCA

KPCA is a statistical model, but all data items are weighted alike. Models in computational linguistics often call for data weighted by frequencies. Weighted KPCA (Wang et al. 2005) is a way to enrich KPCA with frequencies. Under an informative kernel and full rank, WKPCA and KPCA

[1] Or rather its canonical interpretation.

perform equivalently. Using less features (see section 3.2.3) or less dimensions (section 4.1.1) puts WKPCA at a disadvantage. Apparently frequencies bias the model too far towards the frequent cases, such that the model will heavily over-generalize. Compare this to exemplar models (section 2.1.1), where frequent classes are actually *discounted* by means of bias weights on classes. The unweighted model takes all information into account alike, which seems to help in the applications tested here.

8.1.4 Factor rotations and interpretation

Connectionist models trade off performance for interpretability: The closer the model is to human behavior, the more complicated it gets. The high-level terms of the scientific description of the problem become implicit. For example, there is hardly an interpretation to the weights in an artificial neural network, and its nodes are aggregations of information. Nodes usually do not have a single, identifiable function. They only function in conjunction. In designing a network, one usually reserves certain hidden layers to aggregate some parts of the input (see the architecture in MacWhinney et al. 1989), but within these limits one allows for full connection. There is no constraint that weights should be either close to zero or very high.

Such a restriction would be an analogue to rotation techniques in factor analysis (see section 4.2). The overall functionality of the model is distributed differently among its parts (the factors), such that the variance of the feature loadings is maximized. Before that, the factors are just as uninterpretable as nodes in neural network: There are correlations between certain high-level terms and nodes (or principal components), but they are fuzzy. Rotations try to reduce fuzziness, and to focus the hidden factors on a small number of features.

In Kernel PCA, the features are not directly accessible, so this is not an option. Rotation towards data items (with an intended interpretation as prototypes) is also problematic. In chapter 4, I showed how a rotation towards known, hidden features allows constructing a model of data variation with respect to these hidden features. In the example (3.2), this model was of minimal dimension: The three abstract classes fell into a plane. Still these factors were opaque: They are optimized towards performance, not interpretability.

This type of rotation relies on pre-defined classes. As it seems, factor analysis does not substitute the researcher in defining abstractions. It is the ability to abstract over exceptions, to accept worse performance for the sake of a smaller model, that the KPCA models presented here lack. Discarding directions of low explanatory value (see section 4.1) is the KPCA's way of abstraction, but its definition of 'low explanatory value' can only exclude noise, not deliberately reject certain data items as exceptions. The rotation method mentioned above allows identifying negligible dimensions — those that are neutral towards the hidden classes. Again, the researcher's insight comes first.

KPCA models thus are geared towards performance. This is a lot easier to achieve than interpretability, because performance is measurable, and the optimal model is computed exactly (for example as finding maxima in the rotation algorithm by differentiation, see the algorithm in E.2). Formalizing interpretability such that the models can be made (rotated) towards this end is the hard

part.

8.2 Outlook

Considering that there are numerous applications of KPCA and other kernel methods outside linguistics, and only a few (see 2.3) within the field, there are many ways to proceed. I conclude this thesis with an outlook on some exemplary ideas.

8.2.1 Other kernels

n-gram kernels are a good choice within string kernels, when it comes to words, but for sentences substring kernels (n-grams with gaps) are also viable (see Clark et al. 2006). To better capture phonetic similarity, n-grams should not only match when identical, but according to their similarity. See e.g. the *soft matching string kernel* of Shawe-Taylor and Cristianini (2004, p. 370). This similarity is given explicitly by a matrix, or by an inner kernel on phonemes. This is an instance of kernel convolution (Haussler 1999): Similarity is the sum of the similarity of the parts. Shawe-Taylor and Cristianini assume values between 0 and 1; thus a normalized kernel. In preliminary experiments with such a normalized kernel over phonemes (represented as vectors of phonetic features) I found significantly worse results for German plural formation (cf. 6.4.7), probably also due to the normalization step, which hinders exact recognition of features. An intermediate solution would be a kernel over a CV-skeleton: taking all consonants and all vowels as equal. This kernel more easily abstracts over patterns in syllabification etc., but it already encodes linguistic knowledge. A more flexible way would be to consider two phonemes equal if their similarity (as calculated by a kernel over phonemes, see above) exceeds a certain threshold.

I also did not employ kernel addition and multiplication in the examples here. Addition allows combining evidence from several annotation layers, such as gender in plural formation, or semantic similarity in gender prediction (cf. Schwichtenberg and Schiller 2004). Semantic similarity may be specified in a matrix, or it could be calculated, possibly by Latent Semantic Analysis, that is, from vector representations of a word's environment. Multiplication creates pairs of features as the new features. The simplest example is a quadratic kernel: By taking the square of each kernel value, the implicit features are all pairs of the original features. Also, features can be weighted, and so can kernels in kernel addition. In its maximal extension, one can construct kernel polynomials with weighted tuples of features. A kernel polynomial $k^3 + k^2 + k$ has as features all triples and pairs of features of the original kernel k, plus those features themselves. Thus we get exponentially more features with negligible computational cost: The original kernel value has to be derived only once. It remains to be seen whether these tuples of features turn out to be informative, or whether this only leads to too many unseen features. 6-grams, for example, were not always better than 4-grams (see figure 6.2).

For syntactic structures, any of them can be split into parts as in Data-Oriented Parsing (Bod 1998). These parts then form the features of a kernel. Kernels could overcome the main problem of

DOP: the excessive amount of features. See the discussion in 7.3.

8.2.2 Alternative kernel methods

Of course there are Support Vector Machines (Cristianini and Shawe-Taylor 2000), which can be applied to all classification tasks. Among the factor analyses, KPCA is by far not the only one. In the initial review, I mentioned Independent Component Analysis (as employed by Honkela et al. 2010), of which there is also a kernelized variant (Bach and Jordan 2003). The choice of KPCA for this thesis has a touch of arbitrariness. In this thesis, I made particular use of the orthonormal bases it derives, and of its sorting of factors by maximal variance.

The initial sorting is lost during rotation; so as an initial basis any one suffices. It can however be employed to reduce the dimension beforehand, which greatly facilitates rotation. There are more (orthogonal and non-orthogonal) rotations. Future research may reveal rotations leading to an increased linguistic interpretability of factors.

I also explored Weighted KPCA (Wang et al. 2005), but an application where its incorporation of frequencies yields a better model is yet to be found.

8.2.3 Other domains in linguistics

As argued in various places, similarity is best put to use in processes that crucially rely on analogy, such as abstract grammatical features and inflectional/fusional morphology. In this thesis, I have only treated the realization of a single grammatical feature (plural, or past tense); several (e.g. plural and case, or tense and person) can be modeled simultaneously, see the extension of proportional analogy to paradigms (Albright 2008, Pirrelli and Yvon 1999). More structured linguistic objects (more than one morpheme) could be treated step by step; see the iterative method of multi-layer markup in section 5.3.3.

While edge-of-word sentinels are a nice way of pointing the kernel to certain relevant positions in a word, other morphological processes call for other ways of specializing the kernel. As an example, vowel harmony spreading through a word is captured with non-contiguous substrings. Then, sequences of vowels will be implicit features of the kernel. The word space model will only contain vowel sequences obeying the harmony conditions. The general question always is to find the right kernel for a given task. This is done by selecting the right features.

Finally, kernels can be applied wherever explicit feature vectors are in use today. For example, language or author recognition are based on n-grams in practical applications. These tasks and many others can be kernelized.

Appendix

A Kernels

I adapt and extend definitions from Schölkopf and Smola (2002).

(8.1) Variable names:

Definition	Name / Interpretation
R	original space
n	number of data items
$\overrightarrow{a} \in R^n$	data vector
a_i	ith data item
m	number of implicit features
$H := \mathbb{R}^m$	implicit feature space
$\phi\colon R \to H$	implicit mapping
$k\colon R \times R \to \mathbb{R}$	kernel function

Definition 1 (Mapping of data vectors)**.**

$$\forall n \in \mathbb{N}\ \Phi_n\colon R^n \to \mathbb{R}^{m \times n}, (a_k)_{k=1}^n \mapsto (\phi(a_k))_{k=1}^n$$

In an abuse of notation, write $\phi(\overrightarrow{a})$ for $\Phi_n(\overrightarrow{a})$ with the appropriate n. Then, $A := \phi(\overrightarrow{a})$ are the data points in H.

Definition 2 (Kernel)**.** A kernel is an inner product in the feature space, defined via the mapping ϕ:

$$k(a_i, a_j) := \langle \phi(a_i), \phi(a_j) \rangle$$

$\langle \cdot, \cdot \rangle$ is the dot product.

Definition 3 (Injectivity)**.** A kernel is injective iff $\forall x, y \in R\colon \phi(x) = \phi(y) \to x = y$.

Lemma 1 (Distance in the feature space)**.** (Shawe-Taylor and Cristianini 2004, p. 113)

$$\forall x, y \in R\colon \sqrt{|\phi(x) - \phi(y)|^2} = \sqrt{\langle \phi(x) - \phi(y), \phi(x) - \phi(y) \rangle}$$

$$= \sqrt{k(x,x) + k(y,y) - 2k(x,y)} := d(x,y)$$

Definition 4 (Kernel vector)**.**

$$\forall z \in R\colon \overrightarrow{k}(\overrightarrow{a}, z) := \begin{bmatrix} k(a_1, z) \\ \vdots \\ k(a_n, z) \end{bmatrix}$$

Definition 5 (Kernel matrix (or *Gram matrix*))**.**

$$\forall i, j \leq n\colon K_{i,j} := k(a_i, a_j)$$

Or, equivalently: $K := A^T A$

Definition 6 (Positive semi-definiteness)**.** A matrix K is *positive semi-definite* iff

$$\forall \zeta \in \mathbb{R}^n \colon \zeta^T K \zeta \geq 0$$

Lemma 2. Every kernel matrix is positive semi-definite.

Proof. (Shawe-Taylor and Cristianini 2004, p. 57f.)

$$\forall \zeta \in \mathbb{R}^n \colon \zeta^T K \zeta = \zeta^T A^T A \zeta = \|A\zeta\|^2 \geq 0$$

($\|\cdot\|$ is the Euclidean vector norm.) □

Lemma 3. A positive semi-definite matrix K is a kernel matrix.

Proof. (Shawe-Taylor and Cristianini 2004, p. 57f.). A positive semi-definite matrix k has an eigenvalue decomposition $KZ = Z\Delta$. Let $A := \Delta^{\frac{1}{2}} Z^T$. Then:

$$A^T A = Z\Delta^{\frac{1}{2}}\Delta^{\frac{1}{2}} Z^T = Z\Delta Z^T = KZZ^T = KI = K$$

Hence K is a kernel matrix.

□

Lemma 4. A kernel matrix K is symmetric (Schölkopf and Smola 2002, ch. 2).

$$\forall i, j \leq n \colon K_{i,j} = K_{j,i}$$

Proof.

$$\forall i, j \colon K_{i,j} = A_{1:n,i} \cdot A_{1:n,j} = A_{1:n,j} \cdot A_{1:n,i} = K_{j,i}$$

□

A.1 Operations on kernels

A.1.1 Addition

For kernels k_1, k_2 defined on the same domain R, $k_{1+2}(x, y) := k_1(x, y) + k_2(x, y)$ (for all $x, y \in R$) is again a kernel.

A.1.2 Multiplication

For kernels k_1, k_2 defined on the same domain R, $k_{1\times 2}(x, y) := k_1(x, y) \times k_2(x, y)$ (for all $x, y \in R$) is again a kernel.

For further closure properties of kernels see Shawe-Taylor and Cristianini (2004, p. 75).

A.1.3 Normalization

Definition 7 (Normalized kernel)**.** For every kernel $k\colon R \times R \to \mathbb{R}$, there is a (geometrically) normalized kernel $\hat{k} : R \times R \to [0, 1]$ defined as:

$$\forall x, y \in R\colon \hat{k}(x, y) := \frac{k(x, y)}{\sqrt{k(x, x) \cdot k(y, y)}}$$

A.1.4 Kernels over linear combinations

Definition 8 (Linear combination)**.** $\forall \overrightarrow{a} \in R^n, \forall \delta \in \mathbb{R}^n\colon \overrightarrow{a}\delta$ is a linear combination of the data vector.

Definition 9 (Kernel value between an item and a linear combination)**.**

$$\forall x \in R, \forall \overrightarrow{a} \in R^n, \forall \delta \in \mathbb{R}^n\colon k(x, \overrightarrow{a}\delta) := \sum_{i=1}^{n} \delta_i k(x, a_i) = \delta^T \overrightarrow{k}(x, \overrightarrow{a})$$

Lemma 5 (Kernel value between two linear combinations)**.**

$$\forall \overrightarrow{a} \in R^n, \forall \delta, \eta \in \mathbb{R}^n\colon k(\overrightarrow{a}\delta, \overrightarrow{a}\eta) = \sum_{i=1}^{n} \delta_i \sum_{j=1}^{n} \eta_i k(a_i, a_j) = \delta^T K \eta$$

Definition 10 (Difference kernel)**.** Let $\delta = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$.

$$\forall x, y \in R^2\ k_-(x, y) := k(x\delta, y\delta) = k(x_1, y_1) + k(x_2, y_2) - k(x_1, y_2) - k(x_2, y_1)$$

A.2 String kernels

Definition 11 (String)**.** Strings are sequences over an alphabet Σ. Σ^n is the set of all strings of length n. $\Sigma^* := \bigcup_{k=0}^{\infty} \Sigma^k$ is the set of all strings over Σ, $\Sigma^+ \bigcup_{k=1}^{\infty} \Sigma^k = \Sigma^* \setminus \{\epsilon\}$ is the set of all non-empty strings (where ϵ denotes the empty string).

Write $|u|$ $(= n)$ for the length of a string $u \in \Sigma^n$.

Definition 12 (Substring)**.** $u \in \Sigma^n$ is a substring of length n (an n-gram) of $s \in \Sigma^+$ (written $u \sqsubseteq s$) iff $\exists v_1, v_2 \in \Sigma^*\colon s = v_1 u v_2$.[2] The set of all n-grams with $n \leq n_{max}$ is $\Sigma_{n_{max}} = \bigcup_{n=1}^{n_{max}} \Sigma^n$.

Definition 13 (Substring kernel)**.** The substring kernel is defined via its feature mapping $\phi(x) = [\phi_u(x)]_{u \in \Sigma^+}$,[3] where $\phi_u(x)$ counts the occurrences of the substring u in x:

$$\forall u \in \Sigma^+\colon \phi_u(x) = |\{(v_1, v_2)\colon s = v_1 u v_2\}|$$

Replacing Σ^+ with Σ^n yields the n-gram kernel, and replacing it with $\Sigma_{n_{max}}$ yields the up-to-n_{max}-gram kernel.

[2] Juxtaposition of string variables (u, v) denotes concatenation.

[3] This presupposes that the set of strings Σ^n is enumerable. I omit the proof here. The enumeration could follow an ordering first by length and then by Σ.

The n-gram kernel is equivalent to the 'blended spectrum kernel' (Shawe-Taylor and Cristianini 2004, ch. 11.2) with equal weights ($\lambda = 1$) on all spectra. The special case $n = 1$ is called the *Parikh kernel*.

Definition 14 (Subsequence)**.** Let a list of indices $i = (i_1, \ldots, i_n)$ be *ordered* iff $\forall j, k \leq n\colon j < k \Leftrightarrow i_j < i_k$.

$u \in \Sigma^+$ is a subsequence of $x \in \Sigma^+$ (written $u = x(i)$) iff there is an ordered list of indices $i = (i_1, \ldots, i_{|u|})$ (where $|u|$ is the length of u) such that $u_j = x_{i_j}$ for $1 \leq j \leq |u|$.

Definition 15 (Subsequence kernel)**.** Define $\phi_u(x)$ as the number of instances of u as a subsequence of x, and the subsequence kernel via $\phi(x)$ as above (13).

Definition 16 (Gap-weighted subsequence kernel)**.** The gap-weighted subsequence kernel discounts gaps in subsequences by λ ($0 < \lambda \leq 1$):

$\phi_u(x) := \sum_{i\colon u=x(i)} \lambda^{(i_{|u|}-i_1+1)}$ where i ranges over lists of indices.

Algorithm 1 (n-gram kernel)**.**

Input: strings $s, t \in \Sigma^*$, bound n
Output: $k(s, t)$
$z \leftarrow 0$;
foreach $i \in \{1, \ldots, |s|\}$ **do**
  foreach $j \in \{1, \ldots, |t|\}$ **do**
    foreach $k \in \{0, \ldots, min(n-1, |s|-i, |t|-j)\}$ **do**
      if $s_{i+k} = t_{j+k}$ **then** $z \leftarrow z + 1$;
      else break k-loop;
    end
  end
end
return z

For an algorithm for the gap-weighted subsequence kernel, see Shawe-Taylor and Cristianini (2004, Code Fragment 11.3, p. 369).

Lemma 6. The bounded n-gram kernel is not injective.

Proof. Assume a fixed, finite n and an alphabet of at least two distinct symbols: $\{a, b\} \subseteq \Sigma$. Further assume a string $x = a^n b a^n$, and two strings $x_1 = ax$ and $x_2 = xa$. They each contain the n-grams of x and an additional instance of a^n, whether on the left (x_1) or on the right (x_2)

□

A.3 Tree kernels

The tree kernel definitions are adapted from Collins and Duffy (2001b).

Definition 17 (Tree)**.** A tree t is a directed, connected, acyclic and ordered graph with a set of nodes N_t, a set of edges E_t and a unique root $r \in N_t$.

Let $\prec$ be an irreflexive, transitive partial ordering relation on N_t. The order is defined only on nodes which are siblings:

$$\forall t \in T\colon \forall n_1, n_2 \in N_t\colon (n_1 \prec n_2 \vee n_2 \prec n_1) \leftrightarrow \exists n_3 \in N_t\colon (n_3, n_1) \in E_t \wedge (n_3, n_2) \in E_t$$

A labeled tree is a tree with a labeling function $l_t\colon N_t \rightarrow L$, where L is a set of node labels.
A proper tree contains at least two nodes.
A path is a sequence of edges $p = (e_1, \ldots, e_k)$ with edges $e_i \in E_t$ where $\forall i < k\colon e_{i2} = e_{i+1 1}$.
The height of a tree $h(t)$ is defined as the length of the longest path over its edges.
Let equality (=) of trees/nodes be defined as equality of all labels.

In the following, I will assume trees to be proper and labeled. Let T (T_k) be the set of all trees (of height k) over a set of labels L.

Definition 18 (Children of a node)**.**
$\forall t \in T\, \forall n \in N_t\colon ch(n) := \langle n_1, \ldots, n_k \rangle$ where $\forall i \leq k\colon (n, n_i) \in E_t$ and $\forall i, j \leq k\colon i < j \leftrightarrow n_i \prec n_j$

$$\forall t \in T\, \forall n \in N_t\colon nc(n) := |ch(n)| \;\; (= |\{(n, m) \in E_t, m \in N_t\}|)$$

Definition 19 (Subtree)**.** (Collins and Duffy 2001b) u is a subtree of a tree t (written $u \sqsubseteq t$) iff

1. $u \in T$,
2. $N_u \subseteq N_t$,
3. $\forall n \in N_u\colon l_u(n) = l_t(n)$,
4. $\forall n_1, n_2 \in N_u\colon (n_1, n_2) \in E_t \leftrightarrow (n_1, n_2) \in E_u$, and
5. $\forall n \in N_u\colon (\forall m \in N_t\colon (n, m) \in E_t \rightarrow m \in N_u) \vee (\forall m \in N_t\colon (n, m) \in E_t \not\rightarrow m \in N_u)$

u is a proper subtree of t if $u \neq t$.
u is a co-rooted subtree of t if the root of u is the root of t.

Condition 5 is not generally assumed, only in (parts of) linguistics, such as Data-Oriented Parsing (Bod 1998). This difference in definitions is also the reason for the confusion of tree kernel definitions clarified in 7.2.2.1. The present definition is called a *general subtree* in Costa Florêncio 2007, but that name is no clarification either.

Definition 20 (Tree kernel)**.** (From Collins and Duffy 2001a,b; equivalently the *labeled ordered tree kernel* in Costa Florêncio 2007.[4])

[4] Also see the remark in section 7.2.2.1.

Let the indicator function $I_u(n)$ be 1 if the subtree u is rooted at node $n \in N_t$ in a tree t, and 0 otherwise.

Define the tree kernel via its feature mapping ϕ (as in def. 13):[5]

$$\phi_u(t) := \sum_{n \in N_t} I_u(n)$$

Replacing T with T_k yields a height-k tree kernel, and replacing it with $\bigcup_{i=1}^{k} T_i$ yields a height-bounded tree kernel (cf. the substring kernel, def. 13).

Definition 21 (Count of shared co-rooted subtrees)**.**

$$\forall n_1, n_2 \in N \colon C(n_1, n_2) := \begin{cases} 0 \text{ if } n_1 \neq n_2 \vee ch(n_1) \neq ch(n_2) \\ 1 \text{ if } nc(n_1) = 0 \vee nc(n_2) = 0 \\ \prod_{j=0}^{nc(n_1)-1} (1 + C(ch(n_1)_j, ch(n_2)_j)) \text{ else} \end{cases}$$

Algorithm 2 (Tree kernel)**.** This presentation follows Collins and Duffy (2001b); see there for a proof of correctness.

Input: trees $t_1, t_2 \in T$

Output: $k(t_1, t_2)$

$z \leftarrow 0$;

foreach $n_1 \in N_{t_1}$ **do**

  foreach $n_2 \in N_{t_2}$ **do**

    $z \leftarrow z + C(n_1, n_2)$;

  end

end

return z

To implement a bound k on the height of subtrees, replace C with C_k:

Definition 22 (Count of height-bounded shared co-rooted subtrees)**.**

$$\forall n_1, n_2 \in N \colon C_k(n_1, n_2) := \begin{cases} 0 \text{ if } n_1 \neq n_2 \vee ch(n_1) \neq ch(n_2) \vee k = 0 \\ 1 \text{ if } nc(n_1) = 0 \vee nc(n_2) = 0 \\ \prod_{j=0}^{nc(n_1)-1} (1 + C_{k-1}(ch(n_1)_j, ch(n_2)_j)) \text{ else} \end{cases}$$

C can be computed recursively, filling a chart for all pairs $n_1 \in N_{t_1}$, $n_1 \in N_{t_2}$.

[5] Again this presupposes an enumeration. Consider a bracket notation for trees: these are strings, and strings can be enumerated; see fn. 3.

B Kernel Principal Component Analysis

(8.2) Definitions for KPCA:

Definition	Dimension	Name
$v := \frac{1}{n}\mathbf{1}_{n,1}$	$n \times 1$	averaging vector
$V := \mathbf{I}_n - \frac{1}{n}\mathbf{1}_{n,n}$	$n \times n$	centering matrix for K
$M := VA^TAV$	$n \times n$	centered kernel matrix
$M = Q\Delta Q^T$	$n \times n$	eigenvalue decomposition of M
r		rank of M
$Q_r := Q_{1:n,1:r}$		first r eigenvectors
$\Delta_r := \Delta_{1:r,1:r}$		first r eigenvalues
$\alpha_r := Q_r\Delta_r^{-\frac{1}{2}}$	$n \times r$	
$U_r := A\alpha$	$m \times r$	principal components
$\mu := Av$	$m \times 1$	center

Lemma 7. $A^TA = K$, the kernel matrix.

Proof. $[A^TA]_{i,j} = \langle\phi(a_i), \phi(a_j)\rangle = k(a_i, a_j)$ for all $0 < i, j \leq n$. □

Remark 1. If the subscript r is omitted, the full rank of M is implied.

Remark 2. The eigenvectors and eigenvalues in Q resp. Δ are assumed to be in *descending* order.

Definition 23 (Centered)**.** A matrix X's columns (n_X by m_X) are *centered* iff $\mathbf{1}_{1,n}X = \mathbf{0}_{1,m}$. Its rows are centered iff $X\mathbf{1}_{m,1} = \mathbf{0}_{n,1}$. A matrix is centered iff both its rows and its columns are centered.

Algorithm 3 (Kernel Principal Component Analysis (Schölkopf and Smola 2002, p. 431))**.**
Input: kernel matrix K
Output: α, rank
center: $M = VKV$;
diagonalize: $M = Q\Delta Q^T$;
normalize eigenvectors: $\alpha_r = Q_r\Delta_r^{-\frac{1}{2}}$;

B.1 Mappings

Definition 24 (Projection into feature space)**.** An item $z \in R$ is centered and projected onto the first r principal components U_r as follows:

$$f_r(z) := U^T\left(\phi(z) - \mu\right) = (A\alpha_r)^T\left(\phi(z) - Av\right) = \alpha_r^T\left(A^T\phi(z) - A^TAv\right) = \alpha_r^T\left(\overrightarrow{k}(\overrightarrow{a}, z) - Kv\right)$$

Definition 25 (Projecting a point in feature space into the original space)**.**

$$\forall o \in \mathbb{R}^r\colon\ \overrightarrow{a}\zeta := \overrightarrow{a}\alpha_r o + v$$

Lemma 8 (Reconstruction)**.** Insert def. 24 into def. 25.

$$\forall z \in R\colon \overrightarrow{a}\zeta = \overrightarrow{a}\alpha_r f(z) + v = \overrightarrow{a}\alpha_r\alpha_r^T\left(\overrightarrow{k}(\overrightarrow{a}, z) - Kv\right) + v$$

B.2 Hidden features

(8.3) Analogously to m, ϕ, A define:

Definition	Name
l	number of hidden features
$\psi\colon R \to \mathbb{R}^l$	mapping
$B := \psi(\overrightarrow{a})$	matrix of hidden features

Definition 26 (Extraction of hidden features)**.** The *hidden features* of a linear combination $\overrightarrow{a}\zeta$ are given by $B\zeta$.

Lemma 9 (Feature inference)**.** By lemma 8 and def. 26, the vector of hidden features $h(z)$ of $z \in R$ is inferred as:

$$h(z) := B\alpha\alpha^T\left(\overrightarrow{k}(\overrightarrow{a}, z) - Kv\right)$$

B.3 Planar languages

Definition 27 (Planar language; after Clark et al. 2006)**.** $L_0 = \overrightarrow{a}$ is the initial language. The *planar language* L_1 is extrapolated from L_0 (where d is the distance, see lemma 1):

$$L_1 := \{x \in R \mid d(x, \overrightarrow{a}\alpha f(x)) < \epsilon\} \text{ with a threshold } \epsilon$$

C Weighted Kernel Principal Component Analysis

(8.4) Definitions for KPCA with explicit frequencies:

Definition	Dimension	Name
$f \in \mathbb{N}^n$	$n \times 1$	vector of frequencies
$\forall x \in \mathbb{R}\colon f^x\colon \ (f^x)_i = (f)_i^x, 1 \leq i \leq n$	$n \times 1$	pointwise operations on f
$s := \mathbf{1}_{1,n} f$		sum of frequencies
$P\colon P_{i,j} := \begin{cases} 1 \text{ if } \sum_{k=1}^{j-i} f_k < i \leq \sum_{k=1}^{j} f_k \\ 0 \text{ else} \end{cases}$	$n \times s$	expansion matrix (see ex. 2.31 in sec. 2.4.2)
$K' = P^T K P$	$s \times s$	expanded kernel matrix
$V := \mathbf{I}_s - \frac{1}{s}\mathbf{1}_{s,s}$	$s \times s$	centering matrix for K'
$M := VK'V$	$n \times n$	centered expanded kernel matrix
$M = Q\Delta Q^T$	$n \times n$	eigendecomposition of M
$\alpha_r := PQ_r\Delta_r^{-\frac{1}{2}}$	$n \times r$	

Definitions for WKPCA (Wang et al. 2005):

Definition	Dimension	Name
$W := \left(I_{n,n} - \frac{1}{s} f^{\frac{1}{2}} f^{\frac{1}{2}^T}\right) diag(f^{\frac{1}{2}})$	$n \times n$	weighted centering matrix for K
$M_W := WA^T AW^T (= WKW^T)$	$n \times n$	weighted and centered kernel matrix
$M_W = Q_W \Delta_W Q_W^T$	$n \times n$	eigendecomposition of M_W
$\alpha_{Wr} := W^T Q_{Wr} \Delta_r^{-\frac{1}{2}}$	$n \times r$	

Lemma 10. Q_{Wr}'s columns are centered after weighting with $f^{\frac{1}{2}}$.

Proof. (I omit the subscript r.)

$$WKW^T = Q_W \Delta_W Q_W^T \text{ by definition}$$

$$\Rightarrow f^{\frac{1}{2}^T} WKW^T f^{\frac{1}{2}} = f^{\frac{1}{2}^T} Q_W \Delta_W Q_W^T f^{\frac{1}{2}}$$

$$\Rightarrow \mathbf{0}_{1,n} K \mathbf{0}_{n,1} = f^{\frac{1}{2}^T} Q_W \Delta_W Q_W^T f^{\frac{1}{2}} \text{ (by 8.6, p. 171)}$$

$$\Rightarrow \mathbf{0}_{1,1} = \left(f^{\frac{1}{2}^T} Q_W \Delta_W^{\frac{1}{2}}\right)\left(f^{\frac{1}{2}^T} Q_W \Delta_W^{\frac{1}{2}}\right)^T$$

$$\Rightarrow f^{\frac{1}{2}^T} Q_W \Delta_W^{\frac{1}{2}} = \mathbf{0}_{1,r} \text{ by definition of inner products}$$

$$\Rightarrow f^{\frac{1}{2}^T} Q_W = \mathbf{0}_{1,r} \Rightarrow \mathbf{1}_{1,n} diag(f^{\frac{1}{2}}) Q_W = \mathbf{0}_{1,r}$$

□

Lemma 11. $P^{-1}W^TQ_W$ is (column-)centered.

Proof.

$$\mathbf{1}_{1,s}P^{-1}W^TQ_W$$

$$= \mathbf{1}_{1,s}P^T diag(f^{-1})\left(\left(I_{n,n} - \frac{1}{s}f^{\frac{1}{2}}f^{\frac{1}{2}T}\right)diag(f^{\frac{1}{2}})\right)^T Q_W$$

$$= f^T diag(f^{-1})diag(f^{\frac{1}{2}})\left(I_{n,n} - \frac{1}{s}f^{\frac{1}{2}}f^{\frac{1}{2}T}\right)Q_W = \mathbf{1}_{1,n}\left(diag(f^{\frac{1}{2}}) - \frac{1}{s}ff^{\frac{1}{2}T}\right)Q_W$$

$$= \left(f^{\frac{1}{2}T} - \frac{1}{s}\mathbf{1}_{1,n}ff^{\frac{1}{2}T}\right)Q_W = \left(f^{\frac{1}{2}T} - \frac{1}{s}sf^{\frac{1}{2}T}\right)Q_W = \left(f^{\frac{1}{2}T} - f^{\frac{1}{2}T}\right)Q_W = \mathbf{0}_{1,n}Q_W = \mathbf{0}_{1,r}$$

□

Lemma 12. KPCA is a special case of WKPCA, where all frequencies are set to 1.

Proof. Let $f = \mathbf{1}_{n,1}$. $W = I_{s,s} - \frac{1}{s}\mathbf{1}_{n,1}\mathbf{1}_{1,n} = V$. Hence $M = M_W$, therefore $Q = Q_W$ and $\Delta = \Delta_W$, from which follows $\alpha = \alpha_W$.

□

Lemma 13 (Equivalence of expanded and weighted Kernel PCA)**.** Both KPCA on the expanded kernel matrix and the weighted KPCA derive the same set of principal components:

$$\alpha = \alpha_W \text{ and } \Delta = \Delta_W$$

Proof. Both expanded KPCA and WKPCA start with the kernel matrix K, apply some normalization and/or weighting, decompose that matrix, and derive a set of principal components (α, α_W). Deducing $\alpha = \alpha_W$ from $M = Q\Delta Q^T$ and $M_W = Q_W\Delta_W Q_W^T$ is difficult because of the eigenvalue decompositions. Instead, this proof will assume $\alpha = \alpha_W$ and $M_W = Q_W\Delta_W Q_W^T$, and deduce that this defines a Q which is in fact an orthonormal eigenbasis of M (with eigenvalues Δ), and thus the unique eigenvalue decomposition of M.

Assume α_W derived from WKPCA. Then Q is defined as follows:

$$\alpha = \alpha_W \wedge \Delta = \Delta_W \rightarrow PQ_r\Delta_r^{-\frac{1}{2}} = W^TQ_W\Delta_r^{-\frac{1}{2}} \rightarrow PQ_r = W^TQ_W \rightarrow Q_r = P^T diag(f^{-1})W^TQ_W$$

Q and Q_W need to be reduced to the vectors with non-zero eigenvalue, otherwise $\Delta^{-\frac{1}{2}}$ is undefined. Note that no information is lost by this: $Q_W\Delta_W Q_W^T = Q_{W_r}\Delta_{W_r}Q_{W_r}^T$ and $Q\Delta Q^T = Q_r\Delta_r{Q^T}_r$. From here, I omit the subscript r.

$$WKWQ_W = Q_W\Delta_W \text{ by definition of } M_W$$

$$\Rightarrow W^TWKWQ_W = W^TQ_W\Delta_W$$

$$\Rightarrow W^T W K P Q = P Q \Delta_W \text{ substitute } Q_W$$

$$\Rightarrow P^{-1} W^T W K P Q = P^{-1} P Q \Delta_W$$

$$\Rightarrow P^T diag(f^{-1}) W^T W K P Q = Q \Delta_W$$

$$\Rightarrow V P^T diag(f^{-1}) W^T W K P Q = V Q \Delta_W$$

Since $Q = P^T diag(f^{-1}) W^T Q_W$ is centered (lemma 11), is is not affected by centering: $VQ = Q = V^T Q$.

$$\Rightarrow V P^T (I_{n,n} - \frac{1}{s} \mathbf{1}_{n,1} f^T) K P V^T Q = Q \Delta_W \text{ (by 8.7, p. 171)}$$

$$\Rightarrow (V P^T - \frac{1}{s} V \mathbf{1}_{s,1} f^T) K P V^T Q = Q \Delta_W \text{ (by 8.5)}$$

$$\Rightarrow (V P^T - \frac{1}{s} \mathbf{0}_{s,1} f^T) K P V^T Q = Q \Delta_W \text{ (by 8.8)}$$

$$\Rightarrow V P^T K P V^T Q = Q \Delta_W$$

$$\Rightarrow M Q = Q \Delta_W$$

This proves that all vectors in Q are eigenvectors of M. Now I prove that they are orthonormal.

$$Q^T Q = Q_W^T W diag(f^{-1}) P P^T diag(f^{-1}) W^T Q_W$$

$$= Q_W^T W diag(f^{-1}) diag(f) diag(f^{-1}) W^T Q_W$$

$$= Q_W^T \left(I_{n,n} - \frac{1}{s} f^{\frac{1}{2}} f^{\frac{1}{2}^T} \right) diag(f^{\frac{1}{2}}) diag(f^{-1}) diag(f^{\frac{1}{2}}) \left(I_{n,n} - \frac{1}{s} f^{\frac{1}{2}} f^{\frac{1}{2}^T} \right) Q_W$$

$$= \left(Q_W^T - \frac{1}{s} Q_W^T f^{\frac{1}{2}} f^{\frac{1}{2}^T} \right) \left(Q_W - \frac{1}{s} f^{\frac{1}{2}} f^{\frac{1}{2}^T} Q_W \right)$$

$$= \left(Q_W^T - \frac{1}{s} \mathbf{0}_{r,1} f^{\frac{1}{2}^T} \right) \left(Q_W - \frac{1}{s} f^{\frac{1}{2}} \mathbf{0}_{1,r} \right) \text{ by lemma 10}$$

$$= Q_W^T Q_W = I_{r,r} \text{ by orthogonality of } Q_W$$

Hence Q is the (unique) orthonormal eigenvector basis of M, with eigenvalues $\Delta = \Delta_W$.

□

Details glossed over in the derivations above:

$$P^T \mathbf{1}_{n,1} = \mathbf{1}_{s,1} \tag{8.5}$$

$$f^{\frac{1}{2}T} W = f^{\frac{1}{2}T} \left(I_{n,n} - \frac{1}{s} f^{\frac{1}{2}} f^{\frac{1}{2}T} \right) diag(f^{\frac{1}{2}}) = f^T - \frac{1}{s} f^{\frac{1}{2}T} f^{\frac{1}{2}} f^T = f^T - \frac{1}{s} s f^T = \mathbf{0}_{1,n} \tag{8.6}$$

$$\begin{aligned} diag(f^{-1}) W^T W &= diag(f^{-1})(diag(f^{\frac{1}{2}}) - \frac{1}{s} f f^{\frac{1}{2}T})(diag(f^{\frac{1}{2}}) - \frac{1}{s} f^{\frac{1}{2}} f^T) \\ &= (diag(f^{-\frac{1}{2}}) - \frac{1}{s} \mathbf{1}_{n,1} f^{\frac{1}{2}T})(diag(f^{\frac{1}{2}}) - \frac{1}{s} f^{\frac{1}{2}} f^T) \\ &= I_{n,n} - \frac{1}{s} \mathbf{1}_{n,1} f^T - \frac{1}{s} \mathbf{1}_{n,1} f^T + \frac{1}{s^2} \mathbf{1}_{n,1} s f^T \\ &= I_{n,n} - \frac{1}{s} \mathbf{1}_{n,1} f^T \end{aligned} \tag{8.7}$$

$$V \mathbf{1}_{s,1} = (\mathbf{I}_s - \frac{1}{s} \mathbf{1}_{s,s}) \mathbf{1}_{s,1} = \mathbf{1}_{s,1} - \frac{1}{s} \mathbf{1}_{s,s} \mathbf{1}_{s,1} = \mathbf{1}_{s,1} - \frac{1}{s} s \mathbf{1}_{s,1} = \mathbf{1}_{s,1} - \mathbf{1}_{s,1} = \mathbf{0}_{s,1} \tag{8.8}$$

Definition 28 (Extension to the (positive) real numbers)**.** By definition, allow $f \in \mathbb{R}^n_+$.

D Pre-imaging

Pre-imaging is the process of finding an item in input space which best corresponds to a point in feature space (Schölkopf and Smola 2002, ch. 18), according to some kernel k.

Definition 29 ((Approximate) Pre-image; cf. Schölkopf and Smola 2002, p. 547)**.**

$$p(\overrightarrow{a}\,\delta) = \arg\max_{x \in R} \hat{k}(x, \overrightarrow{a}\,\delta)$$

In the general case, there is not necessarily a perfect pre-image $x \in R\colon \hat{k}(x, \overrightarrow{a}\,\delta) = 1$ (Schölkopf and Smola 2002, p. 545).

D.1 Abstract feature pre-imagers

Definition 30 (Pre-image for independent binary features)**.** Count a hidden binary feature as present in a data item $x \in R$ if it exceeds some threshold d, and as absent (0) otherwise.

$$\forall i, 1 \leq i \leq l\colon \psi_i(x) = \begin{cases} 1 \text{ if } \left(h(x)\right)_i > d \\ 0 \text{ else} \end{cases}$$

Definition 31 (Pre-image for abstract features)**.** Assume that the features in ψ are binary and mutually exclusive.

$$\forall i, 1 \leq i \leq l\colon \psi_i(x) = \begin{cases} 1 \text{ if } \forall j \leq l\ (\delta^T B)_{1,i} \geq (\delta^T B)_{1,j} \\ 0 \text{ else} \end{cases}$$

D.2 Generic pre-imaging

R is searched for a pre-image $x_{max} \in R$ iteratively. Starting at an initial item $x_0 \in R$, each step finds the best item x_{i+1} in a set of candidates derived from x_i by a set of structural operations O.

Definition 32 (Structural operation)**.**

$$o : R \to \wp(R)$$

Algorithm 4 (Generic Pre-Imaging)**.**

Input: set of operations O, linear combination $\overrightarrow{a}\,\delta$, initial item x_0

Output: pre-image x_{max}

$x_{max} \leftarrow x_0$;
$d_{max} \leftarrow \hat{k}(x_0, \overrightarrow{a}\,\delta)$;
while *previous* $d_{max} <$ *current* d_{max} **do**
  foreach $x \in \bigcup_{o \in O} o(x_{max})$ **do**
    $d \leftarrow \hat{k}(x, \overrightarrow{a}\,\delta)$;
    if $d > \hat{k}(x_{max}, \overrightarrow{a}\,\delta)$ **then**
      $d_{max} \leftarrow d$;
      $x_{max} \leftarrow x$;
    end
  end
end

D.3 String pre-imager

Definition 33 (String pre-imager operations)**.** Deletions o_d, insertions o_i and substitutions o_s of single characters (cf. Levenshtein distance) are used as operations.

$$\forall x \in \Sigma^+ \colon o_d(x) := \left\{ y \in \Sigma^{|x|-1} \colon y \sqsubseteq x \right\}$$

$$\forall x \in \Sigma^* \colon o_i(x) := \left\{ y \in \Sigma^{|x|+1} \colon x \sqsubseteq y \right\}$$

$$\forall x \in \Sigma^+ \colon o_s(x) := \left\{ y \in \Sigma^{|x|} \colon \exists i \leq |x| \colon x_{1:i-1} x_{(i+1):|x|} = y_{1:i-1} y_{(i+1):|x|} \right\}$$

Operations can be further restricted by allowing only characters to be inserted or deleted the unigram count of which differs between $\overrightarrow{a}\,\delta$ and x_0. This requires making the features of a unigram kernel explicit. The choice of x_0 depends on the task.

E Rotation methods

This section gives definitions for VARIMAX (Kaiser 1958) and rotation towards hidden features (see section 4.2.3). The general algorithm (5) rotates a pair of factors such that a given criterion of variance is maximized. This is done iteratively until the criterion does not improve any more for any pair of factors and any angle. The two methods differ in the criterion they aim to maximize, which determines the calculation of the optimal rotation angle γ_{max}.

Definition 34 (Rotation matrix)**.**

$$R(\gamma) := \begin{bmatrix} cos(\gamma) & -sin(\gamma) \\ sin(\gamma) & cos(\gamma) \end{bmatrix}$$

Algorithm 5 (Rotation algorithm)**.**

Input: factors α, function γ_{max}

Output: rotated factors α

calculate $var(\alpha)$;

repeat

  for *each pair of columns* α_j, α_l **do**

    rotate: $[\alpha_j \alpha_l] \leftarrow [\alpha_j \alpha_l] R(\gamma_{max}(\alpha_j, \alpha_l))$;

  end

until *previous var(*α*)* $\simeq$ *current var(*α*)* ;

A mathematically more precise description would index α for every step of each iteration. Here, I opt for the more legible version, which is also more readily implemented in a procedural programming language, where re-assignment of variables is possible.

E.1 Variation maximization

Definition 35 (VARIMAX criterion (Kaiser 1958))**.**

$$\mu_j := \frac{1}{n} \sum_{l=1}^{n} \alpha_{l,j}^2$$

$$var(\alpha) := \frac{1}{n} \sum_{j=1}^{k} \sum_{i=1}^{n} (\alpha_{i,j}^2 - \mu_j)^2,$$

Kaiser (1958) derives the following variance-maximizing rotation angle for a pair of columns:

$$\gamma_{max}(\alpha_j, \alpha_l) = tan^{-1} \left(\frac{n \sum_{i=1}^{n} 2\alpha_{j,i}\alpha_{l,i}(\alpha_{j,i}^2 - \alpha_{l,i}^2) - \sum_{i=1}^{n}(\alpha_{j,i}^2 - \alpha_{l,i}^2) \sum_{i=1}^{n} 2\alpha_{j,i}\alpha_{l,i}}{n \left(\sum_{i=1}^{n}(\alpha_{j,i}^2 - \alpha_{l,i}^2)^2 - (2\alpha_{j,i}\alpha_{l,i})^2 \right) - \left(\sum_{i=1}^{n} \alpha_{j,i}^2 - \alpha_{l,i}^2 \right)^2 - \left(\sum_{i=1}^{n} 2\alpha_{j,i}\alpha_{l,i} \right)^2} \right) / 4$$

Variance is periodic with respect to γ, with a period of $\pi/2$ (rotating by 90° exchanges the factors and does not change the variance). $\tan^{-1}$ is ambiguous with respect to adding multiples of π. As its outcome is divided by 4, there are two solutions per period; one of them a maximum and the other a minimum, necessarily. A direct comparison is faster than looking into the second derivative.

E.2 Rotation towards hidden features

Definition 36 (Rotation towards hidden features)**.**

$$\mu'_{B,i} := \frac{1}{n}\sum_{l=1}^{n}\left(B^T\alpha_{i,l}\right)^2$$

$$var2_B(\alpha) := \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k}\left(\left(B^T\alpha_{i,j}\right)^2 - \mu'_{B,i}\right)^2$$

This criterion differs from def. 35 in substituting $H^T\alpha$ for α, and in switching rows and columns. We look at pairwise rotated vectors α_x, α_y: $C := B^T\left[\alpha_{1:n,x}\alpha_{1:n,x}\right]R(\gamma)$. Then:

$$\begin{aligned}
& var2'(C) := var2_B(\left[\alpha_{1:n,x}\alpha_{1:n,x}\right]R(\gamma)) \\
= \;& \frac{1}{l}\sum_{i=1}^{l}\sum_{j=1}^{2}\left(C_{i,j}^2 - \frac{1}{2}\sum_{k=1}^{2}C_{i,k}^2\right)^2 = \frac{1}{l}\sum_{i=1}^{l}\sum_{j=1}^{2}\left(C_{i,j}^2 - \frac{C_{i,1}^2 + C_{i,2}^2}{2}\right)^2 \\
= \;& \frac{1}{l}\sum_{i=1}^{l}\sum_{j=1}^{2}\left(\left(\sum_{f=1}^{2}R(\gamma)_{f,j}(B\alpha)_{i,f}\right)^2 - \frac{\left(\sum_{g=1}^{2}R(\gamma)_{g,1}(B\alpha)_{i,x}\right)^2 + \left(\sum_{h=1}^{2}R(\gamma)_{h,2}(B\alpha)_{i,y}\right)^2}{2}\right)^2 \\
= \;& \frac{1}{l}\sum_{i=1}^{l}\sum_{j=1}^{2}\left(\left(R(\gamma)_{1,j}(B\alpha)_{i,x} + R(\gamma)_{2,j}(B\alpha)_{i,y}\right)^2 - \frac{(B\alpha)_{i,x}^2 + (B\alpha)_{i,y}^2}{2}\right)^2 \\
= \;& \frac{1}{l}\sum_{i=1}^{l}\left(\left(cos(\gamma)(B\alpha)_{i,x} + sin(\gamma)(B\alpha)_{i,y}\right)^2 - \mu\right)^2 + \left(\left(-sin(\gamma)(B\alpha)_{i,x} + cos(\gamma)(B\alpha)_{i,y}\right)^2 - \mu\right)^2 \\
= \;& \frac{1}{4l}\sum_{i=1}^{l}(B\alpha)_{i,x}^4 cos(4\gamma) + (B\alpha)_{i,x}^4 + 4(B\alpha)_{i,x}^3(B\alpha)_{i,y}sin(4\gamma) - 6(B\alpha)_{i,x}^2(B\alpha)_{i,y}^2cos(4\gamma) \\
& +2(B\alpha)_{i,x}^2(B\alpha)_{i,y}^2 - 4(B\alpha)_{i,x}(B\alpha)_{i,y}^3sin(4\gamma) + (B\alpha)_{i,y}^4cos(4\gamma) + (B\alpha)_{i,y}^4
\end{aligned}$$

Finding the maximum via differentiation:

$$\begin{aligned}
\textstyle\sum_{i=1}^{l} \quad & -(B\alpha)_{i,x}^4 sin(4\gamma) + 4(B\alpha)_{i,x}^3(B\alpha)_{i,y}cos(4\gamma) + 6(B\alpha)_{i,x}^2(B\alpha)_{i,y}^2sin(4\gamma) \\
& -4(B\alpha)_{i,x}(B\alpha)_{i,y}^3cos(4\gamma) - (B\alpha)_{i,y}^4sin(4\gamma) = 0 \\
\Rightarrow \quad & \gamma_{max\,B}(\alpha_x, \alpha_y) = tan^{-1}\left(\frac{4\sum_{i=1}^{l}(B\alpha)_{i,x}^3(B\alpha)_{i,y} - (B\alpha)_{i,x}(B\alpha)_{i,y}^3}{\sum_{i=1}^{l}(B\alpha)_{i,x}^4 + (B\alpha)_{i,y}^4 - 6(B\alpha)_{i,x}^2(B\alpha)_{i,y}^2}\right)/4
\end{aligned}$$

The remark about periodicity also applies here.

F Naive discriminative learning

(8.9) These definitions are taken from Baayen et al. (2011, appendix), with the notation adapted to match my exposition of KPCA (appendix B).

n	number of words
m	number of cues (surface features)
l	number of outcomes (hidden features)
$\phi(x)$	function from items to cues
$\psi(x)$	function from items to outcomes
A	$m \times n$ matrix of cue occurrences
B	$l \times n$ matrix of outcome occurrences
$f \in \mathbb{N}^n$	vector of word frequencies
$F = diag(f)$	$n \times n$ diagonal matrix of frequencies
$C = AFA^T$	cue-cue co-occurrences
$O = AFB^T$	cue-outcome co-occurrences
$D = diag\left((C_{k,1:m}\mathbf{1}_{n,1})^{-1}_{k=1\ldots m}\right)$	normalization factors
$C' = DC$	cue-cue conditional probabilities
$O' = DO$	cue-outcome conditional probabilities
W	Mixing matrix, via $C'W = O'$

Normalization (conditioning the probabilities) is unnecessary in determining W:

$$W = C'^{-1}O' = (DC)^{-1}DO = C^{-1}D^{-1}DO = C^{-1}O$$

Algorithm 6 (Baayen et al.'s method)**.**
Input: A, B, F
Output: W
solve: $AFA^TW = AFB^T$;

Definition 37 (Mapping)**.** The hidden features of a data item x are calculated as: $\psi'(x) = W^T\phi(x)$
Application to the full data set: $B' = W^TS$

If A is invertible, the mapping is faithful:

$$\begin{aligned} B' &= W^TA = (C^{-1}O)^TA = ((AFA^T)^{-1}AFB^T)^TA = B(AF)^T(AF^TA^T)^{-1}A \\ &= B\underbrace{F^TA^T(F^TA^T)^{-1}}\underbrace{A^{-1}A} = B \end{aligned}$$

G Data

The word lists used in this thesis are all derived from CELEX (Baayen et al. 1996). There are some general encoding issues, and the particularities of the two word lists.

Orthography

All words are cast to lower case for the kernel, but given in actual orthography in here. Equivalently, assume that the kernel considers 'A' and 'a' identical.

Transcription

CELEX comes with four different transcription schemes, all using ASCII characters only. One of them (PhonDISC) is unique in using exactly one character per phoneme. This seems like a good choice for a string kernel, but it also has linguistically undesired side-effects regarding fine-grainedness, such as all tense vowels being necessarily long. All other transcriptions are derived from PhonDISC, so those problems remain. As input to the kernel, I use X-SAMPA notation, but in this thesis I present the words in IPA transcription. Diacritics (`:` for vowel length, ~ for nasality) belong to their head and do not count as separate symbols. Thus ɛ̃ː is a unigram.

Encoding word edges

I encoded all words with edge-of-word sentinels `#`, both for orthography and transcription. `word` thus becomes `#word#`. This allows n-gram kernels to see features such as 'word-initial `w`.

G.1 English Verbs

From CELEX — specifically, the files EMW.cd and EPW.cd (English words, morphology and phonology) — I collected all verbs except auxiliaries and particle and complex verbs (defined as containing a hyphen or whitespace). In the final corpus, there are 5555 regular verbs, 311 irregular verbs, yielding 5866 in total.

Each word has three forms: infinitive/non-3rd-person-singular present, simple past, and past participle; in orthography and transcription. The two derived forms are either marked as irregular, or carry a string denoting the change applied to the word, which in the example given below is read as "subtract y, and add ied".

(8.10)

present	past	participle
occupy	occupied	occupied
	`-y+ied`	`-y+ied`
ˈɒ.kjʊ.paɪ	ˈɒ.kjʊ.paɪd	ˈɒ.kjʊ.paɪd
weave	wove	woven
	irregular	irregular
ˈwiːv	ˈwəʊv	ˈwəʊ.vən
…	…	…

G.2 German Nouns

The set of German nouns is used in the gender (3.2) and plural (6.4) prediction tasks. I extracted all non-compounds with both a singular and a plural form. These are 13873 words. 6642 of them are polymorphemic. The gender distribution is as follows: feminine 47.0%, masculine 36.7%, neuter 16.3%. Some words have more than one gender:

(8.11) m/f Wulst, Topinambur, Spachtel, Ambra, Hode, Karnivore, Batik, Sellerie, Krem, Backhand, Hybride, Haspel, Geisel, Servela. (14)

m/n Meter, Iglu, Indigo, Liter, Input, Radar, Break, Streusel, Avis, Knaul, Tetrachord, Ocker, Bonbon, Makadam, Rhododendron, Indult, Essay, Portable, Tentakel, Knäuel, Nugat, Schlamassel, Kratt, Anastigmat, Sylvester, Paravent, Schelf, Joghurt, Aquädukt, Silo, Trajekt, Spind, Schemen, Spier, Rebus, Zoophyt, Klafter, Sims, Teil, Kardamom, Output, Yoghurt, Dekor, Spart, Supremat, Spray, Bloch, Gummi, Silvester, Tarock, Virus, Ridikül, Tabernakel, Willkommen, Azimut, Tingeltangel, Ar, Match, Gulasch, Semaphor, Flakon, Katapult, Frottee. (62)

f/n Säumnis, Elastik, Nocturne. (3)

There is only one word which can have any gender: “Dingsbums” (*thingamajig*). However, it does not have a plural, and is thus excluded.

The format of an entry is shown in 8.12. For a discussion of the morphological information, see section 3.2.1.2.

(8.12)

singular	plural	gender	morphology
Schrift	Schriften	fem.	schreib-
ˈʃrɪft	ˈʃrɪf.tən		

Frequency data is obtained from the field ‘Mannheim written frequency’, which is based on a 5.4 million word newspaper corpus. In all applications, I added 1 to each frequency to smooth out zero frequencies.

G.2.1 Gender assignment

This is the list of manually evaluated test items for the experiment in 3.2.2. The KPCA model is compared to the cues of Köpcke (1988). The last column gives the cue used to determine gender. 'default' refers to the many phonological cues for masculine.

Table G.1: Comparison of gender assignment

word	gender	KPCA model	rules	cue
intuition	f	f	f	morph
strebung	f	f	f	morph
husar	m	m	m	phon
taschenspielerei	f	f	f	morph
generosität	f	f	f	morph
quote	f	f	f	-e
halter	m	m	m	morph
pelz	m	n	m	phon
ertränkung	f	f	f	morph
absurdität	f	f	f	morph
rötling	m	m	m	morph
erblindung	f	f	f	morph
gastronom	m	m	m	default
beuge	f	f	f	-e
lein	m	n	m	default
toxikum	n	n	n	morph
bänkchen	n	n	n	morph
trünklein	n	m	n	morph
empfindlichkeit	f	f	f	morph
depot	n	m	m	phon
wollkämmer	m	m	m	morph
balsa	n	m	m	default
manko	n	m	m	default
haft	m??	f	f	phon
versäumnis	n	n	f/n	ambiguous
orientale	m	m	f	-e
einzeichnung	f	f	f	morph
bestrafung	f	f	f	morph
linguist	m	m	m	morph
vinkulierung	f	f	f	morph
sträßchen	n	n	n	morph
minderheit	f	f	f	morph

Table G.1: Comparison of gender assignment

word	gender	KPCA model	rules	cue
studium	n	n	n	morph
fuchtel	f	n	m	phon
akromegalie	f	f	f	morph
elektrotechniker	m	m	m	morph
tartrat	n	m	m/n	ambiguous
edikt	n	n	m	phon
kapelan	m	f	m	default
filigran	n	m	m	default
wall	m	m	m	phon
rekrut	m	m	m	default
entschiedenheit	f	f	f	morph
natter	f	m	m	default
feinheit	f	f	f	morph
integral	n	m	m	default
schirmung	f	f	f	morph
abkömmling	m	m	m	morph
verweigerer	m	m	m	morph
xylometer	n	m	m	default
fragwürdigkeit	f	f	f	morph
bereinigung	f	f	f	morph
trimmung	f	f	f	morph
renunziation	f	f	f	morph
karikatur	f	f	f	morph
evangelium	n	n	n	morph
reiterin	f	f	f	morph
synesis	f	n	f	morph
anschluß	m	m	m	phon
punch	m	m	m	phon
schauder	m	n	m	default
senke	f	f	f	-e
giraffe	f	f	f	-e
tampen	m	m	m	default
geheimnis	n	n	f/n	ambiguous
beleuchtung	f	f	f	morph
shampoo	n	m	m	default
dichte	f	f	f	-e
okkupant	m	m	m	morph

Table G.1: Comparison of gender assignment

word	gender	KPCA model	rules	cue
legalisierung	f	f	f	morph
pogrom	n	m	m	default
bühne	f	f	f	-e
spezifität	f	f	f	morph
perversität	f	f	f	morph
aufseherin	f	f	f	morph
kohle	f	f	f	-e
niednagel	m	f	m	phon
stabilisator	m	m	m	morph
sekretär	m	n	m/n	ambiguous
fender	m	m	m	default
aufkauf	m	m	m	phon
pilgrim	m	m	m	default
storch	m	m	m	phon
herbst	m	n	m	phon
leber	f	m	m	default
sukzession	f	f	f	morph
rächer	m	m	m	morph
holzung	f	f	f	morph
metaller	m	m	m	morph
seismometrie	f	f	f	morph
zentner	m	m	m	default
salami	f	m	m	default
bolschewik	m	m	m	default
korpus	m	m	m	morph
nationalisierung	f	f	f	morph
kommutation	f	f	f	morph
spray	n	m	m	phon
aufrechnung	f	f	f	morph
kontakt	m	m	m	default
krach	m	m	m	phon
correct	100	77	81	

Bibliography

Adam Albright. Islands of reliability for regular morphology: Evidence from Italian. *Language*, 78 (4):684–709, 2002.

Adam Albright. Modeling analogy as probabilistic grammar. Unpublished manuscript, Massachusetts Institute of Technology, Cambridge, MA, 2008.

Adam Albright and Bruce Hayes. Modeling english past tense intuitions with minimal generalization. In *Proceedings of the ACL-02 workshop on Morphological and phonological learning - Volume 6*, MPL '02, pages 58–69, 2002.

Adam Albright and Bruce Hayes. Rules vs. analogy in English past tenses: a computational/ experimental study. *Cognition*, 90(2):119–161, 2003.

Doug Arnold and Evita Linardaki. A data-oriented parsing model for HPSG. In Anders Søgaard and Petter Haugereid, editors, *2nd International Workshop on Typed Feature Structure Grammars (TFSG'07)*, pages 1–9, 2007.

F. Gregory Ashby and Leola A. Alfonso-Reese. Categorization as probability density estimation. *Journal of Mathematical Psychology*, 39(2):216–233, 1995.

R. Harald Baayen. Corpus linguistics and naive discriminative learning. *Brazilian Journal of Applied Linguistics*, submitted.

R. Harald Baayen, R. Piepenbrock, and L. Gulikers. CELEX2. Technical report, Linguistic Data Consortium, University of Pennsylvania, 1996.

R. Harald Baayen, Petar Milin, Dusica Filipović Đurđević, Peter Hendrix, and Marco Marelli. An amorphous model for morphological processing in visual comprehension based on naive discriminative learning. *Psychological Review*, 118:438–482, 2011.

Francis R. Bach and Michael I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2003.

Dagmar Bittner and Klaus-Michael Köpcke. Acquisition of the german plural markings. a case study in natural and cognitive morphology. In Chris Schaner-Wolles, John Rennison, and Friedrich Neubarth, editors, *Naturally! Linguistic studies in honour of Wolfgang U. Dressler presented on the occasion of his 60th birthday*, pages 47–58. Rosenberg & Sellier, 2001.

Rens Bod. *Beyond Grammar - An Experience-Based Theory of Language*. CSLI Publications, 1998.

Rens Bod and Ronald Kaplan. A DOP Model for Lexical-Functional Grammar. In Rens Bod, Remko Scha, and Khalil Sima'an, editors, *Data-Oriented Parsing*, pages 211–233. Stanford, California: CSLI Publications, 2003.

Gosse Bouma. Finite state methods for hyphenation. *Natural Language Engineering*, 1:1–16, 2002.

John A. Bullinaria. Modelling, reading, spelling and past tense learning with artificial neural networks. *Brain and Language*, 59:236–266, 1997.

Razvan C. Bunescu and Raymond J. Mooney. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 724–731, Stroudsburg, PA, USA, 2005a. Association for Computational Linguistics.

Razvan C. Bunescu and Raymond J. Mooney. Subsequence kernels for relation extraction. In *Proceedings of the 19th Conference on Neural Information Processing Systems (NIPS'05)*, 2005b.

Basilio Calderone. Unsupervised decomposition of morphology. a distributed representation of the italian verb system. In *Proceedings of the Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR 2008)*, pages 82–89, 2008.

Steve Chandler. The english past tense: Analogy redux. *Cognitive Linguistics*, 21(3):371–417, 2010.

Tat-Jun Chin and David Suter. Incremental Kernel Principal Component Analysis. *IEEE Transactions on Image Processing*, 16(6):1662–1674, June 2007.

Alexander Clark, Christophe Costa Florêncio, and Chris Watkins. Languages as hyperplanes: Grammatical inference with string kernels. In *Proceedings of the European Conference on Machine Learning (ECML)*, pages 90–101, 2006.

Michael Collins and Nigel Duffy. Parsing with a Single Neuron: Convolution Kernels for Natural Language Problems. Technical report, University of Santa Cruz, 2001a.

Michael Collins and Nigel Duffy. Convolution Kernels for Natural Language. In *Proceedings of Neural Information Processing Systems (NIPS 14)*, 2001b.

Michael Collins and Nigel Duffy. New ranking algorithms for parsing and tagging: kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 263–270, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

R. Cook, P. Kay, and T. Regier. The world color survey database: History and use. In H. Cohen and C. Lefebvre, editors, *Handbook of Categorisation in the Cognitive Sciences*, pages 223–242. Elsevier, 2005.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20, 1995.

Christophe Costa Florêncio. Tree planar languages. In *Seventh IEEE International Conference on Data Mining Workshops*, pages 405–410, 2007.

Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

Nello Cristianini, John Shawe-Taylor, and Huma Lodhi. Latent semantic kernels. *Journal of Intelligent Information Systems*, 18:127–152, 2002.

Aron Culotta and Jeffrey Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

Walter Daelemans. A comparison of analogical modeling of language to memory-based language processing. In R. Skousen, D. Lonsdale, and D. Parkinson, editors, *Analogical Modeling*, pages 157–179. John Benjamins, 2002.

Walter Daelemans and Antal van den Bosch. Generalization performance of backpropagation learning on a syllabification task. connectionism and natural language processing. In *Proceedings of the Third Twente Workshop on Language Technology*, pages 27–38, 1992.

Walter Daelemans, J. Zavrel, K. van der Sloot, and Antal van den Bosch. Timbl: Tilburg memory based learner, version 6.0, reference guide. *ILK Technical Report Series*, 07-05, 2007.

David Danks. Equilibria of the rescorla-wagner model. *Journal of Mathematical Psychology*, 47(2): 109–121, 2003.

Ferdinand de Saussure. *Cours de Linguistique Generale*. Paris: Payot, 1916.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

David Eddington. Analogy and the dual-route model of morphology. *Lingua*, 110(4):281–298, 2000.

Caroline Féry. German word stress in optimality theory. *The Journal of Comparative Germanic Linguistics*, 2:101–142, 1998.

Ulrike Hahn and Ramin Charles Nakisa. German inflection: single route or dual route. *Cognitive Psychology*, 41:313–360, 2000.

David Haussler. Convolution kernels on discrete structures. Technical report, University of Santa Cruz, 1999.

Timo Honkela and Aapo Hyvärinen. Linguistic feature extraction using independent component analysis. In *In Proceedings of IJCNN'04*, pages 279–284, 2004.

Timo Honkela, Aapo Hyvärinen, and Jaakko J. Väyrynen. WordICA — emergence of linguistic representations for words by independent component analysis. *Natural Language Engineering*, 16(3):277–308, 2010.

A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. New York: John Wiley & Sons, 2001.

Gerhard Jäger. Natural color categories are convex sets. In Maria Aloni, Harald Bastiaanse, Tikitu de Jager, and Katrin Schulz, editors, *Logic, Language and Meaning*, volume 6042 of *Lecture Notes in Computer Science*, pages 11–20. Springer Berlin / Heidelberg, 2010.

Gerhard Jäger. Using statistics for cross-linguistic semantics: a quantitative investigation of the typology of color naming systems. *Journal of Semantics*, to appear. URL `http://www.sfs.uni-tuebingen.de/~gjaeger/publications/josColorsFinal.pdf`.

Frank Jäkel, Bernhard Schölkopf, and Felix A. Wichmann. Generalization and similarity in exemplar models of categorization: Insights from machine learning. *Psychonomic Bulletin & Review*, 15 (2):256–271, 2008.

Frank Jäkel, Bernhard Schölkopf, and Felix A. Wichmann. Does cognitive science need kernels? *Trends in Cognitive Sciences*, 13(9), 2009.

Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Machine Learning: ECML-98*, volume 1398 of *Lecture Notes in Computer Science*, pages 137–142. Springer Berlin / Heidelberg, 1998.

I.T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, 1986/2002.

Henry Kaiser. The varimax criterion for analytic rotation in factor analysis. *Psychometrika*, 23(3): 187–200, 1958.

Hisashi Kashima and Teruo Koyanagi. Kernels for Semi-Structured Data. *ICML '02: Proceedings of the 19th International Conference on Machine Learning*, pages 291–298, 2002.

Rohit Jaivant Kate. *Learning for Semantic Parsing with Kernels under Various Forms of Supervision*. PhD thesis, The University of Texas at Austin, 2007.

Emmanuel Keuleers. *Memory-based learning of inflectional morphology*. PhD thesis, Universiteit Antwerpen, 2008.

Emmanuel Keuleers and Dominiek Sandra. Similarity and productivity in the english past tense. *Transformation*, 234:1–49, 2003.

Emmanuel Keuleers, Dominiek Sandra, Walter Daelemans, Steven Gillis, Gert Durieux, and Evelyn Martens. Dutch plural inflection: the exception that proves the analogy. *Cognitive Psychology*, 54(4):283–318, June 2007.

Toshi Konishi. The semantics of grammatical gender: A cross-cultural study. *Journal of Psycholinguistic Research*, 22:519–534, 1993.

Klaus-Michael Köpcke. Schemas in German plural formation. *Lingua*, 74(4):303–335, April 1988.

Klaus-Michael Köpcke and David A. Zubin. Zubin & Köpcke 1984 affect classification.pdf. *Lingua*, 63:41–96, 1984.

Vanja Kovic, Gert Westermann, and Kim Plunkett. Implicit vs. explicit learning in German noun plurals. *Psihologija*, 41(4):387–411, 2008.

Krista Lagus, Mathias Creutz, and Sami Virpioja. Latent linguistic codes for morphemes using independent component analysis. In *Ninth Neural Computation and Psychology Workshop: Modeling Language, Cognition and Action*, 2004.

Thomas K. Landauer and Susan T. Dumais. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104(2):211–240, 1997.

Joseph P. Levy, John A. Bullinaria, and Malti Patel. Explorations in the derivation of semantic representations from word co-occurrence statistics. *South Pacific Journal of Psychology*, 10:99–111, 1998.

F.M. Liang. *Word Hy-phen-a-tion by Com-put-er*. PhD thesis, Stanford University, 1983.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text Classification using String Kernels. *Journal of Machine Learning Research*, 2(3):419–444, August 2002.

Kevin Lund and Curt Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behaovior Research Methods, Instruments, & Computers*, 28(2):203–208, 1996.

Brian MacWhinney, Jared Leinbach, Roman Taraban, and Janet McDonald. Language learning: Cues or rules? *Journal of Memory and Language*, 28(3):255–277, 1989.

M. P. Maratsos and M.A. Chalkley. The internal language of children's syntax: The ontogenesis and representation of syntactic categories. *Children's Language*, 2, 1980.

Gary F. Marcus, Ursula Brinkmann, Harald Clahsen, Richard Wiese, and Steven Pinker. German inflection: The exception that proves the rule. *Cognitive Psychology*, 29:189–256, 1995.

John McCarthy and Alan Prince. Foot and word in prosodic morphology: The arabic broken plural. *Natural Language and Linguistic Theory*, 8:209–282, 1990.

J. L. McClelland and Karalyn Patterson. Rules or connections in past-tense inflections: What does the evidence rule out. *Trends in Cognitive Sciences*, 6:465–472, 2002.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. Semantic role labeling via tree kernel joint inference. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 61–68, Stroudsburg, PA, USA, 2006.

J. Mugdan. *Flexionsmorphologie und Psycholinguistik*. Tübingen: Narr, 1977.

Ramin Charles Nakisa and Ulrike Hahn. Where defaults don't help: The case of the german plural system. In *Proceedings of the 18th Annual Meeting of the Cognitive Science Society*. Erlbaum, 1996.

Vivi Nastase and Marius Popescu. What's in a name? in some languages, grammatical gender. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1368–1377, 2009.

Robert M. Nosofsky. Attention, similarity, and the identification-categorization relationship. *Journal of Experimental Psychology: General*, 115:39–57, 1986.

Robert M. Nosofsky and Roger D. Stanton. Speeded classification in a probabilistic category structure: Contrasting exemplar-retrieval, decision-boundary, and prototype models. *Journal of Experimental Psychology: Human Perception and Performance*, 31(3):608–629, 2005.

Robert M. Nosofsky and Safa R. Zaki. Exemplar and prototype models revisited: Response strategies, selective attention, and stimulus generalization. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 28(5):924–940, 2002.

Steven Pinker and Alan Prince. n language and connectionism: Analysis of a parallel distributed processing model of language acquisition. *Cognition*, 28:73–193, 1998.

Steven Pinker and Michael T. Ullman. The Past-Tense Debate: The past and future of the past tense. *Trends in Cognitive Sciences*, 6(11):456–474, November 2002.

Vito Pirrelli and Francois Yvon. The hidden dimension: a paradigmatic view of data-driven NLP. *J. Exp. Theo. Artif. Intel.*, 11:391–408, 1999.

Kim Plunkett and Patrick Juola. A Connectionist Model of English Past Tense and Plural Morphology. *Cognitive Science*, 23(4):463–490, October 1999.

Kim Plunkett and Ramin Charles Nakisa. A Connectionist Model of the Arabic Plural System. *Language and Cognitive Processes*, 12(5-6):807–836, October 1997.

Alan Prince and Paul Smolensky. Optimality theory: Constraint interaction in generative grammar. Technical report, Rutgers University and University of Colorado at Boulder, 1993/2004. ROA 537, 2002. Revised version published by Blackwell, 2004.

Michael Ramscar. The influence of semantics on past-tense inflection. In *Proceedings of the 23rd Annual Conference of the Cognitive Science Society*, 2001.

Michael Ramscar. The role of meaning in inflection: Why the past tense does not require a rule. *Cognitive Psychology*, 45:45–94, 2002.

D. E Rumelhart and J. L. McClelland. On learning the past tenses of english verbs. In D. E Rumelhart and J. L. McClelland, editors, *Parallel distributed processing*, pages 216–271. MIT Press, 1986.

Helmut Schmid, Bernd Möbius, and Julia Weidenkaff. Tagging syllable boundaries with joint n-gram models. In *INTERSPEECH-2007*, pages 2857–2860, 2007.

Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. Adaptive Computation and Machine Learning series. MIT Press, 2002.

Bernhard Schölkopf, Alexander J. Smola, and Klaus-Robert Müller. Kernel principal component analysis. In Wulfram Gerstner, Alain Germond, Martin Hasler, and Jean-Daniel Nicoud, editors, *Artificial Neural Networks — ICANN'97*, volume 1327 of *Lecture Notes in Computer Science*, pages 583–588. Springer Berlin / Heidelberg, 1997.

Beate Schwichtenberg and Niels O. Schiller. Semantic gender assignment regularities in german. *Brain and Language*, 90:326–337, 2004.

John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

R. N. Shepard. Toward a universal law of generalization for psychological science. *Science*, 237: 1317–1323, 1987.

Niels A. Taatgen. Extending the past-tense debate: a model of the german plural. In J.D. Moore and K. Stenning, editors, *Proceedings of the twenty-third annual conference of the cognitive science society*, pages 1018–1023. Erlbaum, 2001.

Niels A. Taatgen and John R. Anderson. Why do children learn to say "Broke"? A model of learning the past tense without feedback. *Cognition*, 86(2):123–55, 2002.

Peter Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188, 2010.

Antal van den Bosch. Expanding k-nn analogy with instance families. In Royal Skousen, Deryle Lonsdale, and Dilworth B. Parkinson, editors, *Analogical modeling, an exemplar-based approach to language*. Amsterdam: John Benjamins, 2002.

Jaakko J. Väyrynen and Timo Honkela. Comparison of independent component analysis and singular value decomposition in word context analysis. In Honkela, Könönen, Pöllä, and Simula, editors, *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation (AKRR)*, pages 135–140, 2005.

Yannick Versley, Alessandro Moschitti, Massimo Poesio, and Xiaofeng Yang. Coreference systems based on kernels methods. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 961–968, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

A. Wagner and R. Rescorla. A theory of pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In A. H. Black and W.F. Prokasy, editors, *Classical conditioning ii*, pages 64–99. Appleton-Century-Crofts, 1972.

Fei Wang, Jingdong Wang, and Changshui Zhang. Spectral feature analysis. *IJCNN*, 2005.

W. A. Wickelgren. Context-sensitive coding, associative memory and serial order in (speech) behavior. *Psychological Review*, 76:1–15, 1969.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106, March 2003.