Statistical Language Models for Alternative Sequence Selection

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan Tilburg University,
op gezag van de rector magnificus,
prof. dr. Ph. Eijlander,
in het openbaar te verdedigen ten overstaan van een
door het college voor promoties aangewezen commissie
in de aula van de Universiteit
op woensdag 7 december 2011 om 18:15 uur

door

Johan Herman Stehouwer
geboren op 16 juni 1983 te Delfzijl

Promotores: Prof. dr. A.P.J. van den Bosch
Prof. dr. H.J. van den Herik
Copromotor: dr. M.M. van Zaanen
Beoordelingscommissie:
Prof. dr. E.J. Krahmer
Prof. dr. E.O. Postma
Prof. dr. W. Daelemans
Prof. dr. C. de la Higuera
Prof.dr. F.M.G. de Jong

Netherlands Organisation for Scientific Research

The research reported in this thesis has been funded by NWO, the Netherlands Organisation for Scientific Research in the framework of the project *Implicit Linguistics*, grant number 277-70-004.

SIKS Dissertation Series No. 2011-45
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

TiCC Ph.D. Series No. 19.
ISBN 978-94-6191-049-3

Cover artwork & design by Levi van Huygevoort (levivanhuygevoort.com)
Printed by IPSKamp, Enschede

Preface

In 2006 I decided to take the opportunity to become a Ph.D. student at the Tilburg University. I was given the task to study some challenging problems in computational linguistics and to improve my research skills. I was happy and grateful that such an avenue for self improvement was open to me, and I still consider it a privilege that I have had the opportunity to work on a single topic relatively free of worries for a good four years. I encountered a variety of interesting problems and learned more about my field and myself than I could have foreseen.

First and foremost I would like to thank my supervisors Antal van den Bosch, Jaap van den Herik, and Menno van Zaanen. Their endless enthusiasm and feedback shaped me as I took my first steps as a researcher. Antal has always given me free reign for pursuing my own research interests. Jaap has spent hours and hours helping me sharpen my academic writing and argumentation skills. Menno has provided me with daily suggestions and feedback on the details of my work; moreover he helped me to define the shape of the complete picture.

The research was performed at the Tilburg center for Cognition and Communication (TiCC). TiCC is a research center at the Tilburg School of Humanities (TsSM). The research presented in this thesis was performed in the context of the Implicit Linguistics project. I would like to acknowledge the Netherlands Organisation for Scientific Research (NWO) for funding this project as part of the Vici program. The research reported in this thesis has been carried out under auspices of SIKS, the Dutch School for Information and Knowledge Systems. SIKS was acknowledged by the Royal Dutch Academy of Arts and Sciences (KNAW) in 1998.

My life at the university has been interwoven with the coffee club. I would like to thank Menno and Jeroen for their enthusiastic support of our passion: good coffee. This coffee club was for me a stimulating source of fresh ideas, both academic and silly.

The time spent on my Ph.D. was an exciting time. It has been successful thanks to the colleagues at the Tilburg University, in particular those of the ILK group. We had coffee breaks, ILK-Barbies, and Guitar Hero parties as well as the occasional foray into the interesting world of Belgian beers. The atmosphere on the third floor of the Dante building was always helpful, friendly, and supportive. All other persons not mentioned above who helped me in the various stages of my research are equally thanked for their willingness. In summary, I would like to thank all the fine colleagues at ILK, TiCC, Dante, and outside Dante. Thank you very much for making my time enjoyable.

Finally, I would like to thank the people closest to me. My thanks go out to my close friends, with whom I have had some great times. For instance, we had some spiffy trips, pool games, and nights at the pub. Then, special thanks go to my parents who have always been supportive and who have pushed me to pursue everything to the best of my abilities. Of course, Laurence, you know best what we endured. You encouraged me to continue and to spend many nights writing this thesis and in these times you remained so supportive. Thank you.

Herman Stehouwer June 2011

Contents

Chapter 1

Introduction

Natural language processing (NLP) is a field which is part of both computer science and linguistics. It regards the processing of natural language with the help of computers. Three language processing tasks that use NLP techniques are spelling correction, machine translation, and speech recognition. A well established research direction approaches these tasks in NLP using language models. Most typically, a language model determines how well a sequence of linguistic elements fits the model, which by extension provides an estimate of how likely the sequence is in its language. In our examination we focus on statistical language models. The objective of this thesis is to investigate whether adding explicit linguistic information to these language models leads to better results when processing text, using the assumption that the given information may be already implicitly present in the text.

The current chapter introduces the reader to the topic under consideration and provides the essential elements for our investigation. In Section 1.1 we describe statistical language models that support us in a variety of tasks. Section 1.2 discusses the process of selecting the best alternatives out of a set of alternative sequences, each of which represents a possible change to the underlying text. In Section 1.3 we formulate our problem statement, followed by the formulation of four research questions in Section 1.4. Section 1.5 gives our research methodology. Finally, Section 1.6 provides the structure of the thesis.

1.1 Statistical Language Models

In NLP, language models are used for many different tasks. For most of the tasks, having access to a language model is essential for performing the task

well. A language model can be used to decide on whether, or to what degree, a sequence belongs to the language. A statistical language model (SLM) is a language model that is characterised by a variety of distributions over (parts of) the language. The distributions are measured and lead to statistics on distributions and probabilities on sequences. Next to the task of (1) estimating the likelihood of a sequence, a language model may also be used for two other tasks: (2) to generate language sequences, and (3) to decide between different alternative sequences.

Applications of Language Models

Language models are typically used at places where a decision has to be made on the suitability of a member from a set of possible sequences. Five example applications that effectively use language models are speech recognition, machine translation, optical character recognition, text generation, and text correction. Below, we briefly describe the five applications and the use of statistical language models for these tasks.

Speech recognition deals with recognising a spoken utterance from its corresponding audio stream. Statistical language models are used to select a sequence from all the possible spoken sequences of words that fits the observed data best. The task for language models in speech recognition is exacerbated by the fact that the audio stream has many possible interpretations, both in segmentation and in the selection of the individual phonemes. The interpretations result in a large lattice of possible words and sequences matching the sound, out of which a selection will have to be made.

Machine translation deals with translating a text written in one language to a text written in another language. It tries to find a mapping of (parts of) a sequence to another sequence. The possible mappings can result in many candidate translations. Typically, translating one sentence results in a choice between hundreds or several orders of magnitude more candidate translations, where candidates differ in the order and choice of words or subsequences. Statistical language models contribute to a score by which the best translation can be selected. Several factors make machine translation hard. We mention three of them: (1) the word order of the languages involved can be different; (2) the number of words for a correct translation can be smaller or larger than the number of words in the original sequence; and (3) the other language might not have the same translation for each sense of the word (e.g., the word *bank*[1] in English might refer to several wholly different meanings each with their own translation possibilities).

[1] We use *italics* to denote an example.

Optical character recognition deals with converting a text in an image into machine-readable text. Mapping a picture of some characters into the actual characters is hard to do automatically and often leads to multiple possible character sequences. Statistical language models are used to select between possible different alternative conversions of words in a context in combination with statistical models of character images and the context of those characters.

Text generation deals with generating a new text with a pre-defined meaning. Specific applications that make use of text generation are machine translation and the generation of text starting from concepts; for instance, the generation of a weather forecast from forecast data. When formulating a text, there are usually many possible correct formulations. Statistical language models are used to select between several generated options.

Text correction deals with a set of problems that involve modifying (erroneous) text. We experiment with problems taken from the domain of text correction when investigating the effects of different language models. The main idea is that the text is transformed into a better text in the same language. Statistical language models are used here to choose a solution between different alternative rewordings.

Problems in Text Correction

As stated above we concentrate on problems related to the task of text correction. Here we introduce the topic globally by mentioning and briefly describing four problems out of the large variety of problems in text correction, namely (1) non-word error correction, (2) confusible error detection and correction, (3) verb and noun agreement correction, and (4) prenominal adjective reordering. These problems come with their own set of challenges and possible approaches. Below we give a flavour of these challenges.

Non-word error correction deals with the detection and correction of non-word errors in a text. A non-word error is the occurrence of a malformation of a word, caused by the insertion, deletion or substitution of one or more letters of the original word, resulting in a character string (token) that is not a proper word in the language. The task is then to map this malformed word to the correct word if such a correct word can be determined.

One of the difficult factors is the fact that not all words unknown to the system are malformed. In any unseen text, the system is bound to observe words it has not seen before, for instance, because of the creative aspects of productive morphology. A typical example of this would be noun compounding, as found in amongst others Dutch, German, and Afrikaans. Two examples are the

Dutch word *fietspomphouder* (bicycle pump holder), and *dozenstapel* (stack of boxes).

An example of a non-word error is **Only one woord is wrong.*[2] In the example, the word *woord* is not a correct English word. A possible correction could be to replace it by the word *word*. For more information, techniques, and correction methods, the interested reader is referred to Kukich (1992) and Reynaert (2005).

Confusible error detection and correction is an extensively studied topic. A confusible error occurs when a word is confused with a different word which is incorrect within the context. For example, in the sentence **There car is red* the word *There* should be replaced by *Their*. Several forms of relatedness give rise to confusible errors. Words that are mutually confused, and form a confusible set, can be related by many factors, including homophony, similar writing, keyboard proximity, and similar morphology. An example of a confusible set based on similarity in phonemes is {*then, than*}.

Verb and noun agreement correction deals with the detection and correction of agreement errors of verbs and nouns in relation to their sentential context. These types of errors occur when the verb and noun are incongruent with their context, e.g., the number or the gender of the subject and verb do not match. An example of an agreement error in a full sentence would be **The man run*. This example can be corrected by either changing the noun to *men* or the verb to *runs*. In languages other than English the situation can be much more complex.

Prenominal adjective reordering deals with the correct ordering of prenominal adjectives. This problem does not always have a clear, best solution. Then again, language users do have preferences for certain orderings. An example is ?? *the wooden red cabin*[3] versus *the red wooden cabin*.

The Use of n-grams

A statistical language model is trained on a collection of language sequences before it is used. Such a collection of sequences cannot be complete, as it is impossible by any standards to enumerate all sequences that are possible in a natural language. Confronted with a new sequence, it is unlikely that the model has encountered this sequence in advance. Yet, it is likely that the model has encountered subsequences of the sequence.

---

[2] The character * announces an erroneous example. We mark the entire sentence that contains the error.

[3] From here on we will mark all less preferred examples that are not, strictly speaking, incorrect by a double question mark. This is also in line with the approach used in the literature, specifically in Malouf (2000).

Statistical language models assign likelihood scores to sequences. A score is usually given by combining scores of fixed-size parts of the sequence. These fixed-size parts are called n-grams. An n-gram is a set of n consecutive symbols as they occur in a sequence. We note that historically the term n-gram denotes sequences of n characters. When we refer to n-grams in this thesis, we refer to sequences of n words. Statistical language models use distributions over n-grams to assign the scores to sequences.

Fixed-size n-grams provide for a limited flexibility, which is a serious obstacle for language models. That is, the size of the n-grams used in the language model is predetermined, i.e., not flexible. For instance, a 4-gram model will not store all permissible combinations of four words, it will only store the combinations of four words that it has seen. Therefore, a statistical language model will often store n-grams of smaller sizes as *back-off* sub-models. It means that when the data for a certain distribution (e.g., 4-grams) does not contain the 4-gram in question, a back-off step is made to a distribution (e.g., 3-grams) for which relatively more data is available (see Subsection 4.1.3). This is one of the ways to try and deal with sparseness. Back-off increases the applicability of the statistical language model at the expense of extra storage.

An important obstacle for statistical language models is *sparseness*. It means that for certain distributions, such as those of 4-grams or 5-grams there is insufficient data for an accurate approximation of the real distribution. This implies that the model will be confronted with valid sequences in which n-grams occur that the model has not encountered before. Sparseness is unavoidable for complex sequences such as sentences in natural language. We return to this topic in Chapter 4.

Our aim is to develop a statistical language model that is able to select the best solution among a number of alternatives. We use suffix arrays (see Section 3.5) to implement this language model because we can use them to approach the data flexibly (e.g., for calculating probabilities for n-grams of any size). Suffix arrays allow us to look up the positions and number of occurrences of any subsequence of the training material efficiently, enabling us to calculate probabilities for any sequence (as far as it occurs in the training material). When calculating n-grams of any size we are no longer bound by the fixed predetermined maximum size. For a standard language model, we are able to determine, on a case-by-case basis, the largest n-gram match that is still useful, i.e., that is associated with a non-null probability.

1.2 Alternative Sequence Selection

In the context of our research, a statistical language model is used to select between sets of possible alternatives. The alternative that fits the statistical language model best is selected. Note that in Latin, *alter* means the other one of two. In the Dutch language, some still attach this meaning to the word *alternatief* (alternative). We use the English interpretation, meaning that alternative is different from the original (in Latin, *alius*).

As we stated earlier, statistical language models face the obstacle of sparseness. Besides n-gram back-off, an alternative to mitigating the sparseness issue is by using word-level annotations. These annotations provide additional information when the statistical language model is unable to derive reliable statistics from the words in the sequence. With annotations, we generally mean linguistically motivated annotations that denote certain linguistic abstractions that can be assigned to words, such as their morpho-syntactic function in the particular sequence. In this way the annotation can act as a back-off. In the field of NLP a great deal of attention is given to annotations of natural language texts. The underlying idea of such annotations is that they help the computer with NLP tasks, as they offer linguistically usable generalisations. In some NLP tasks, for example, it is useful to know that a newly encountered word, for which no statistical information is available, is probably a noun. The statistics that can be gathered for categories such as noun are an aggregate of the statistics of all nouns, which is likely to be useful back-off information for any new noun.

Two general classes of annotations of words in a sequence can be distinguished. The first class is the class of human-designed annotations, where the annotation is designed in advance inspired by explicit linguistic insights. The second class is the class of machine-derived annotations. The computer uses the inherent structure of the text to create annotations that denote certain structures suggested by, e.g., information-theoretical analyses. We remark that the class of human-designed annotations is usually applied to new text using supervised machine learning systems, i.e., incorporating linguistic knowledge in a learning system. In contrast, machine-derived annotations are applied to new text using unsupervised machine learning systems, i.e., incorporating knowledge in a learning system that is derived from the data only.

Several types of linguistically motivated annotations of natural language texts exist. Two of the best known are part-of-speech tags and the grammatical parse structure, such as dependency parses. Part-of-speech tags denote for each word its category within the sequence based on its syntactic and/or morphological function. Typically, these annotations are assigned by hand to a reference corpus. Computer systems may be used to assist in this process (Van den Bosch 2009). From a reference corpus, usually manually annotated, automatic sys-

tems can be trained that apply the annotations to new, previously unseen, material.

The grammatical parse structure is an annotation that denotes the syntactic structure of a sentence. It describes the relations between words (and groups of words). This is contrast to part-of-speech tags which deal with the function of a single word. As with part-of-speech tags these annotations are typically assigned by hand to a reference corpus.

In this thesis we aim to investigate the effect of these annotations on the statistical language model. We can study the effects of the annotation on alternative sequence selections using language models. The annotation for the alternatives can be determined relatively straightforwardly on texts for which an annotation exists. In order to study the effects of the addition of an annotation, we restrict ourselves to possible alternatives where the changes in the sequence are localised. Thus, as a baseline evaluation method, we make local changes to a sequence to generate the alternatives. We are then able to check whether our language model selected the original sentence as the most likely sequence among the alternatives.

Earlier we mentioned three problems within the task of text correction that conform to these restrictions, viz. confusible correction, verb and noun agreement correction, and prenominal adjective ordering selection. They can all be approached by changing small, localised parts of a larger sequence. In this thesis we do not investigate correction of errors, instead we look at the selection of the original sequence from a set of alternative sequences. With the four other tasks mentioned, the differences between alternative sequences are much larger, making the specific effects harder to study. For (1) automatic speech recognition, the assignment of annotations is made difficult by the alternatives possibly constituting completely different sentences. For (2) machine translation, the order of the words in the target sequence as well as the words to use in the target sequence are not fixed, resulting in many possibly radically different alternative sequences. For (3) optical character recognition, the surface form of the words is often malformed making the assignment of annotations difficult to do, prior to correction of the OCR-ed text. For (4) text generation, the differences between the sequences that can be generated with the same or a similar meaning are also not localised.

1.3 Problem Statement

As a key challenge of statistical language modeling, we identify the sparseness problem. We identify two possible solutions to mitigate the problem. Namely,

(1) the flexibility of the n-gram, and (2) the use of annotations. Below we provide a summary and a line of reasoning for choosing these two.

First, we approach the issue of flexibility of language models. Often, statistical language models split the input into equal size n-grams of words. They do so in order to make assigning probabilities to sequences tractable. We will investigate how a flexible size of n-gram, with or without annotation, impacts the processing of a set of alternative sequences. There will be parts in a sequence that are more regular than other parts. In the more regular parts, it should be possible to come closer to the ideal model, namely that of evaluating the complete sequence.

Second, we approach the issue of annotations on data. For statistical language modelling, we rely on data. If the data is insufficient we stumble into the problem of sparseness. Sometimes annotations may offer some relief. The study of annotations is a world in itself. It contains a rich diversity of approaches. For an adequate overview we refer to Garside et al. (1997), Kingsbury et al. (2002), and Van Eynde (2004).

Expert-based, linguistically motivated annotations, such as part-of-speech tags and dependency parses are learnable by a computer system when a pre-annotated corpus is available. Once learned they can be automatically applied to new text.

After assigning annotations to alternative sequences, the ability of the language model to choose among alternative sequences should improve as the annotations should help to access more reliable statistics.. The annotations can help combat the effects of the sparseness problem by providing a different back-off possibility.

The production of human-designed annotation schemes and hand-annotated corpora are expensive in terms of expert or annotator time. The schemes and annotated corpora can be used (and are needed) for training automatic systems that apply those annotations. To resolve these practical issues, machine-derived (or unsupervised) annotation schemes and the corresponding automatically annotated corpora may provide some relief. They can be seen as an alternative to human-designed annotations.

Based on the above observations, our problem statement (PS) consists of two parts (PS 1 and PS 2). They read as follows.

Problem Statement 1. (Flexibility) *Is it helpful to create a statistical language model that is flexible, i.e., not fixed in advance, with regards to the n-gram size, for adequately handling the problem of sparseness?*

Problem Statement 2. (Annotations) *Do linguistically motivated annotations and their automatically generated counterparts provide information that can*

be successfully used as a back-off step to handle sparseness? Does alleviating sparseness in this way increase performance on alternative-sequence-selection tasks?

1.4 Research Questions

To answer the two-fold problem statement, we developed four research questions (RQs). To answer problem statement 1 we developed RQ1 and RQ2. To answer problem statement 2, we developed RQ3 and RQ4. Below we provide background and reasons for our research questions.

Flexibility

We remark that most of the current language models are not flexible; they are frequently limited to predetermined-size n-grams. Admittedly, fixed-size n-grams make the problem of assigning probabilities to sequences more tractable. Limiting n helps dealing with sparseness by only examining a small part of the sequence at hand. In our opinion, it is desirable to have a system which is more flexible. So, we aim at a system that can deal with flexible size n-grams, i.e. a system for which the size of the n-gram is not predetermined. Thus, our first research question reads as follows.

Research Question 1. *Is there a need to predetermine or limit the size of the n-grams used in language models? Is there an inherent advantage or disadvantage to using a fixed-size n?*

Some tentative considerations on RQ1 follow below. If the n-gram size is larger, sparseness will have a greater impact. The sparseness of n-grams becomes more of an issue for each increase of n. If the n of the n-gram is no longer limited to a fixed value, how do we deal with the greater sparseness of the larger n-grams? Can we sidestep or avoid this obstacle?

When the n-gram size is too large, the calculation of the probability becomes impossible due to sparseness. However, for each sequence the size of the n-gram at which the calculation becomes impossible will be different. The largest n-gram size that can be used can be determined by examining the distributions at a position in a sequence.

Research Question 2. *If the size of the n-grams is not fixed in advance, how can we still generate comparable distributions when we select among alternative sequences?*

Annotations

To aid the computer in dealing with the task of alternative sequence selection, the text may be used with annotation. Here we remark that, similar information to what the annotations model is also derivable, to some extent, from the un-annotated text given a discovery procedure. It is an open question whether we need these annotations for a decision in alternative sequence selection. If we assume that the information is already implicitly available in the un-annotated text. Therefore, we will investigate whether annotations improve the performance of the language model.

Research Question 3. *Is there a benefit to including annotations in the language model, measured as a better performance on alternative sequence selection tasks?*

Research Question 4. *Is there a difference in performance on alternative sequence selection tasks when using human-designed annotations compared to machine-generated annotations?*

machine-derived annotations may provide a level of abstraction similar to linguistically motivated ones.. It makes information explicit that is already implicitly available in the data. By contrasting the two types of annotations we will examine the differences. Appropriate answers to these four RQs will help us to answer both PS1 and PS2.

1.5 Research Methodology

The thesis investigates how we can make language models flexible with regards to (a) not predetermining the size of the n-gram and (b) supporting annotations. We focus on three key issues of this flexibility: (1) the impact of a flexible-size n-gram-based language model, (2) the impact on the back-off capabilities of a language model with respect to added annotation information, and (3) contrasting the difference between using manually created annotations versus machine-derived annotations.

This section outlines the research methodology that describes the approach we have used for answering PS1 and PS2. The methodology consists of four parts, viz. literature review and analysis, designing and using an experimental platform, measuring the effects of flexibility, and evaluation. They are briefly discussed in the subsections below.

Literature Review and Analysis

Literature review is at the base of our research into language models for alternative sequence selection. The literature review concentrates on: (1) literature on language modelling, specifically n-gram-based language modelling (we will also mention other types of language models); (2) literature on the essential tasks for which we use alternative sequence selection; (3) literature on manually created annotations and machine-derived annotation schemes and annotations.

Designing and Using an Experimental Platform

In order to deploy flexible size n-grams and annotations in language models, there is a need for a flexible experimental platform. The design, development, and use of the platform concentrates on three key issues: (1) a separate language-model component that is easily modified in order to change major aspects of the language model; (2) an alternative-sequence-generation component; and (3) a clear data model shared by all parts of the platform. We aim at obtaining results while keeping the model as constant as possible on different alternative-sequence-selection tasks.

Measuring the Effect of Flexibility and Annotations

Below we mention how we perform our measurements. The effects we want to measure in order to answer the problem statements are partitioned into three phases.

In the first phase we will introduce a flexible n-gram-based language model (without any annotation). This model will not be tied to a predetermined size or predetermined sizes of n-grams. We will try to scale the n-gram in order to come closer to the ideal of evaluating the sequence as a whole. This flexible system will be used for the three tasks of alternative sequence selection.

In the second phase we will investigate the effects of local annotations on the performance of the three tasks. To investigate the effects of the different annotations we will add them to the language model in two ways. First, we will add human-designed assigned part-of-speech annotations. Second, we will add a machine-derived annotation that approximates the part-of-speech annotations.

In the third phase we investigate the use of syntactic dependency annotation. First, we will add human-designed dependency-parse annotations. Second, we will add machine-derived annotations. The machine-derived annotations try to establish a dependency structure automatically. The dependency structure is

sentence-global, in the sense that the dependencies between words can span the entire sentence.

Evaluation

We will study the effects of (1) the changes to the language model, and (2) the changes to the data used, by evaluating the performance of three alternative-sequence-selection tasks. The evaluation will be partitioned into three parts, analogous to the introduction of the effects studied.

The sub-division will be as follows: (1) we evaluate the effects of variable size n-grams without any annotation; (2) we evaluate the addition of human-designed and machine-derived annotations at a local level; and (3) we evaluate the addition of human-designed and machine-derived annotations at a global level.

The evaluation is performed by investigating how well the statistical language models predict the gold standard. From the corpus (the gold standard) we derive alternative sequence sets, on which we perform a selection step using the language model. When the model predicts the sequence that is also in the gold standard, this is counted as correct even if the gold standard would at points be considered incorrect upon inspection. We remark that the gold standard contains very few errors.

We provide an example of the non-word error rate of a frequently used corpus. In an error analysis of running text of the Reuters RVC1 corpus, Reynaert (2005) found that 21% of all types were non-word errors. As most of these types have low frequencies, the errors account for one error in every four hundred tokens, i.e., 0.25% of the text.

1.6 Structure of the Thesis

The structure of the thesis is as follows. In Chapter 1, we introduce the reader to the topic and provide the essential elements for our investigation. Moreover, we formulate two problem statements with each two research questions and we outline our research methodology.

In Chapter 2, we describe three problems that serve our investigation throughout the thesis. The three alternative-sequence-selection problems are used to study the impact of several design choices for the language model. In Chapter 3, we introduce the experimental setup that we use throughout the thesis.

In Chapter 4, the first language-model type is discussed. We examine a language model which is not limited in the size of the n-gram used. The language model is employed without any annotation. We try to answer RQ1 and RQ2, and thereby PS1. Chapter 5 deals with the second language-model type. The language models belonging to this type are enriched with local annotations dependent on the local context of the word. This chapter provides partial answers to RQ3 and RQ4, on the basis of which we address PS2. In Chapter 6 we discuss the third type of language models. They are enriched with annotations dependent on a global context. This chapter provides additional answers to RQ3 and RQ4, on the basis of which we address PS2.

In Chapter 7 we provide answers to the RQs, to PS1 and PS2 and formulate our conclusions. Furthermore, we describe future work that can be pursued on the basis of this thesis.

Chapter 2

Three Alternative Selection Problems

The thesis focusses on three alternative selection problems. The three alternative selection problems are taken from the domain of text correction. Our choice is guided by the expectation that the solutions to these problems require only a localised change to the structure of the sequence in order to make the sequence correct. The three problems allow us to study the effects of changes to the model on a set of well-understood and studied issues.

The chapter introduces the three problems. Each of them provides a different element in our investigation. In Section 2.1 we describe the problem of confusibles. In Section 2.2 we discuss the problem of verb and noun agreement. Then, in Section 2.3 we treat the problem of prenominal adjective ordering.

2.1 Confusibles

The confusible problem deals with the *detection* and *correction* of confusibles. A confusible occurs when a word is confused with a different word which is incorrect within the given context. Confusible errors are quite common. There are a number of different approaches to identify confusible sets (see Subsection 2.1.1). They are described in this section together with several approaches to handle the selection of the correct element within the elements of the confusible set (see Subsection 2.1.2). Within the context of this thesis we will restrict ourselves to the confusible sets as used by Golding and Roth (1999). These confusible sets have been used in many experiments and publications.

They allow us to study the effects of making language models more flexible with regards to (1) the n-gram size and (2) the presence or absence of annotations.

The use of a predefined set of confusibles has as a consequence that identification of confusibles is not considered as part of the problem under investigation. For investigations of the identification of confusibles we refer to an article published by Huang and Powers (2001).

Several forms of relatedness give rise to confusible sets. A confusible set is typically quite small (consisting of two or three elements). In the literature we found four forms of relatedness. A word can be related by sound, similar writing, keyboard proximity, and similar meaning. For instance, when word forms are homophonic, they often tend to become confused in writing (cf. the pronunciations of *to*, *too*, and *two*; *affect* and *effect*; or *there*, *their*, and *they're* in English) (cf. Sandra et al. 2001, Van den Bosch et al. 2007).

Below we briefly discuss a confusible set taken from the list by Golding and Roth (1999). We consider its relations with other areas of research, and mention two facets of the problem. Our example is the set {*then*, *than*}. This confusible set accounts for a part of confusible errors mostly made by non-native speakers of English. A straightforward erroneous example using this set is given in Example 1.

Example 1. * *The cat is smaller then the dog.*

In Example 1, the only possible correction is replacing *then* by the word *than*. The error can only be corrected by looking at the sentential context.

The confusible problem can be seen as a specialised form of the all-word prediction task (Van den Bosch 2006a). In all-word prediction, the task is to predict the next word in a sequence or the word that should be inserted in a gap in a sequence. A unique characteristic of the word prediction task, compared to many other natural language tasks, is that real-world training examples are abundant. Any digitally available text can be used as training material.

Confusibles are an active area of research. Many different approaches have been proposed. The approaches largely differ in two facets: (1) the *identification* of possible confusible sets, and (2) the *selection* of the correct alternative given a confusible set. *Selection* between members of a confusible set (i.e., alternatives) in the context given leads directly to *correction* if the selected alternative is different from the original. We discuss both facets separately below.

2.1.1 Identification of Confusible Sets

An important step in the approach to identify confusibles is the definition of possible confusible sets. Each possible confusible set represents a set of words which account for a part of the errors in a sequence. For the identification of such sets we distinguish two approaches: (1) the manual selection of confusible sets, such as done by Golding and Roth (1999), and (2) the automatic creation of confusible sets based on a similarity metric such as in Huang and Powers (2001).

Two Identification Approaches

The first identification approach[1] of confusible sets is by Golding and Roth (1999) who wrote a seminal work on confusible correction. Their main contribution is a classification-based system for spelling correction. Their point of departure was the manual creation of confusible sets. They used 21 different confusible sets taken from the introduction of a dictionary in three different categories of confusion: (1) closeness in phonetic distance, (2) closeness in spelling distance, and (3) relatedness in meaning[2]. We remark that the notion of closeness implicitly defines the notion of distance. The distance between the surface form of words and the distance between the representation of words[3] can be expressed by the Levenshtein (1966) distance metric.

A different focus is proposed by Banko and Brill (2001). They only studied the effects on two confusible sets: (1) {*then, than*}, and (2) {*among, between*}. They used the corresponding confusible task, and employed it to study the effects of scaling to large training corpora.

The second identification approach is by Huang and Powers (2001) who primarily identified confusible sets that are close in measurable distance. They computed the distance between words using two representations: (1) the distance on the keyboard between words, and (2) the distance between words when represented using a phonetic representation. For both these representations they modelled the insertion, deletion, and substitution operations. An example for the keyboard proximity (we refer here to a qwerty-type keyboard) would be that the letter *d* might be replaced by the letters *s, w, e, r, f, v, c*, or *x*. Huang and Powers (2001) also identified a third type of confusibles, namely (3) suspect words (errors) made by second language learners (found in databases

[1] Which is not really an identification approach as it used a static, human-defined list of confusible sets.

[2] We note that there is some overlap between categories. For instance {*affect, effect*} is close in both phonetic and spelling distance.

[3] For instance, phonological representation using the IPA alphabet.

and corpora). All the sets of words that are identified using the metrics (1), (2), and (3) were stored as confusible sets.

2.1.2 Selection of the Correct Member

Almost all approaches in the literature use a classifier to perform the confusible selection step. A classifier is a machine learning system that predicts a class based on an input vector (also known as an instance). In this case the input is a representation of the sentential context (including the focus confusible). Typically, a single classifier is trained for each set of confusibles.

For the selection of the correct member of an identified confusible set, a variety of different classifiers have been used in the approaches mentioned above. Golding and Roth (1999) use a layered classifier approach. Banko and Brill (2001) use four different selection approaches: (1) a memory-based classifier, (2) a WINNOW-based classifier (analogous to Golding and Roth 1999), (3) a naive-bayes-based classifier, and (4) a perceptron-based classifier. Huang and Powers (2001) used statistics on local syntactic patterns in order to select between the elements of the confusible set.

Most work on confusibles using machine learning concentrates on hand-selected sets of notorious confusibles. The machine learner works with training examples of contexts containing the members of the confusible set (cf. Yarowsky 1994, Golding 1995, Mangu and Brill 1997, Wu et al. 1999, Even-Zohar and Roth 2000, Banko and Brill 2001, Huang and Powers 2001, Van den Bosch 2006b).

2.2 Verb and Noun Agreement

Verb and noun agreement is a classical problem in natural language processing. It concerns the question whether a verb or a noun agrees with the rest of the sentence. The literature on this classical problem mainly focusses on learning English as a second language (ESL). ESL corpora are typically rich sources of errors.

Below we discuss two aspects of this task as they occur in the literature: (1) the identification of disagreement in the sequence (see Subsection 2.2.1), and (2) the selection of a correct word at an identified position (see Subsection 2.2.2)

2.2.1 Identification of Agreement

The verb and noun agreement problem deals with the congruence of verbs and nouns in a sentence. If a noun or verb does not agree in one of its aspects within its corresponding sentence there is an incongruence. Below we show two examples (see Examples 2 and 3).

Example 2. * *He sit at the table.*

In Example 2 we show a sentence that is incongruent in the verb *sit*. If we replace *sit* by the word *sits* it would result in a correct sequence. We remark that changing the subject of the sentence to *They* would also be a valid correction.

Example 3. * *The trees burns.*

In Example 3 we show an example of a sentence that is (a) incongruent in the noun *trees*, or (b) incongruent in the verb *burns*. If we replace *trees* by the word *tree* it would result in a correct sequence. Changing the verb to *burn* would also result in a correct sequence.

Six Identification Approaches

Many different approaches to correcting the verbs and nouns with the goal of making the sequence congruent have been examined. Below we briefly discuss six approaches to identifying incongruence: (1) the n-gram-based approach, (2) the mutual-information-based approach, (3) the semantic-relatedness approach, (4) the canonical-form approach, (5) the labelled-sequential-patterns approach, and (6) the rich-feature-selection approach. For each approach, we cite a typical publication, preferably the original.

The first approach is the n-gram-based approach. One of the earliest n-gram-based approaches is that by Mays et al. (1991). The n-gram-based approaches use sequences of n-grams to estimate, locally, whether a different word would fit better in that context, if that word resembled the word to be replaced. It is mentioned by Mitton (1996) as one of the few known systems in 1996 that attempts to handle real-word errors. Wilcox-O'Hearn et al. (2008) reconsidered the approach and compared it with a WORDNET (Miller et al. 1990) based approach as described by Hirst and Budanitsky (2005).

The second approach is based on Mutual Information (MI)[4]. Chodorow and Leacock (2000) present ALEK, a system for detecting grammatical errors in

[4] Mutual information is a measure of mutual dependence of two variables. We return to the MI metric in detail in Subsection 5.1.3.

text. It uses negative evidence for the combination of target words collected from a secondary, small training corpus. Interesting is the use of the MI metric, comparing the probability of the occurrence of bi-grams to that of the product of the probabilities of the corresponding uni-grams. Chodorow and Leacock assume that when a sequence is ungrammatical the MI metric should be negative. Negative values occur when the co-occurrence of the bigram is very unlikely compared to the product of the occurrences of the corresponding unigrams. It means that normally in a running text the MI metric remains positive, but when the metric drops to a negative value for a certain position the system has detected a potential error and will try to replace the word in question.

The third approach is an approach based on semantic relatedness of words as described by Hirst and Budanitsky (2005). For each word in the text that is suspect (e.g., a noun or a verb) they investigate the existence of a semantic relatedness between that word and its sentence. If such a relatedness is not found, spelling variants are examined for semantic relatedness. If a variant is related, it is suggested to the user as a possible replacement. The measure of relatedness is based on WORDNET (Miller et al. 1990). Suspect words are all words occurring in the text which (1) do not occur elsewhere in the text, (2) are not part of a fixed expression, and (3) are not semantically related to the nearby context.

The fourth approach is based on the canonical form and developed by Lee and Seneff (2006). A sentence is stripped down to its *canonical form* before being completely rebuilt. In the canonical form all articles, modals, verb auxiliaries, and prepositions are removed, and nouns and verbs are reduced to their stems. They create a lattice of all possibilities (i.e., all possible articles, modals, and verb auxiliaries at each position) at all positions and then traverse the lattice using the Viterbi (1967) algorithm. Lee and Seneff (2006) try several different ranking strategies for re-ranking the possible alternatives including a word tri-gram model and a parser. The parser is used with a domain specific context-free grammar, trained on the training set. Later, Lee and Seneff (2008) focussed specifically on verb form correction using the parser and tri-gram approach mentioned.

The fifth approach is by Sun et al. (2007). They approach the problem of errors in second-language-learner sequences by learning labelled sequential patterns. Pattern discovery is done using correct and incorrect examples from Chinese-written and Japanese-written English language corpora. These patterns represent a typical part of a type of erroneous sequence, such as $<$*this, NNS*$>$ [5] (e.g., contained in * *this books is stolen.*).

[5] Here, *NNS* is a part-of-speech tag that stands for plural noun. For part-of-speech tags *NN* is often used as the tag for the nouns and *NNS* for the plural nouns. Examples that contain such a tag are the Penn Treebank tagset and the CLAWS 5 tagset.

The sixth approach is developed by Schaback and Li (2007). They use co-occurrence, bigrams, and syntactic patterns to serve as features for a support vector machine classifier. They outperform the systems they compare against[6] through the features used on the recall[7] measure. However, on precision[8] Schaback and Li (2007) are outperformed by all compared systems except ASPELL.

2.2.2 Selection of the Correct Agreement

Most approaches in the literature use a classifier to perform the selection step. For verb and noun agreement a generative approach is also taken from time to time. For instance, Lee and Seneff (2006) use such a generative approach.

For the selection of the correct word form for the verb or the noun, a variety of approaches are used in the literature mentioned above. Mays et al. (1991) perform selection by using a tri-gram model and selecting the most likely sequence. Chodorow and Leacock (2000) only detect errors, without selecting a correct alternative. Hirst and Budanitsky (2005) use the same method for selection as for detection. They order possible corrections by their semantic relatedness and the most related possibility was selected. The selection criterium that Lee and Seneff (2006) use is based on the language model used, the best traversal through the lattice is selected as the correct sequence. Sun et al. (2007) does not try to select a correction candidate. Schaback and Li (2007) uses a support vector machine classifier to select a correction candidate.

2.3 Prenominal Adjective Ordering

Prenominal adjective ordering is a problem that has been mostly studied in the linguistics literature. The ordering of the prenominal adjectives is important for the fluency of the resulting sentence. As such it is an interesting candidate for computational approaches. Naïve computational attempts (e.g., using a simple bigram model) already attain a fairly high performance of around 75% prediction accuracy on newspaper texts. Malouf (2000) has improved this result to around 92% by adequately putting partial orderings to use. The per-

[6] Schaback and Li (2007) compare their system to the following systems: MS WORD, ASPELL, HUNSPELL, FST, and GOOGLE. We remark that ASPELL does not take context into account, so it is not surprising that it is outperformed.

[7] Recall is used to measure the coverage of a system. In this case it denotes the percentage of faults present in the data that where found by the system.

[8] Precision is used to measure the correctness of a system. In this case it denotes the percentage of correctly spotted faults compared to the total number of potential faults indicated.

formance measures do not take into account different adjective orderings that occur for reasons of focus or contrast, i.e., they are counted as not preferred even if the author intended that specific ordering. In this section, we discuss the investigation of the order (Subsection 2.3.1) and the selection procedure (Subsection 2.3.2).

2.3.1 Investigation of the Ordering

Below we give two alternative examples of prenominal adjective orderings. The first is preferred over the second.

Example 4. *the large wooden red cabin*

Example 5. *?? the red wooden large cabin*

In Example 4 we see the following order: size, material, colour. In Example 5 we see: colour, material, size. A correction system should not place a hard constraint on these orders. In practice, some orderings are less correct, e.g., the one shown in Example 5 is not preferred compared to Example 4. However, some orderings are more 'correct' than others. Therefore, the order of prenominal adjective modifiers is a challenge with a subtle preference system.

The investigation of the order is studied by many linguists in countries from all over the world. Language users have their own background, culture, and taste. Within the field of linguistics the ordering is also a debated issue.

Feist (2008) devoted his thesis to this problem of prenominal adjective ordering. He included a thorough overview of the relevant literature. He stated about the literature the following.

> Views on English premodifier order have varied greatly. They have varied as to whether there are distinct positions for modifiers or a gradience, and as to the degree and nature of variability in position. (Feist 2008, p. 22)

For discussion of linguistic issues we refer to Feist (2008). Below we deal with computational approaches.

2.3.2 Selection: Seven Computational Approaches

Below we discuss two studies that together contain seven computational approaches of the prenominal adjective ordering problem. Specifically, the studies try to find the best ordering of a set of (at least two) prenominal adjectives.

This problem has most notably been studied in a computational manner by Shaw and Hatzivassiloglou (1999), and Malouf (2000). Both publications deal with finding evidence for orderings where the absolute best order is not known. Still, if evidence for a possible order is found in the data it is taken into account.

Shaw and Hatzivassiloglou (1999) presented a system for ordering prenominal modifiers. The authors propose and evaluate three different approaches to identify the sequential ordering among prenominal adjectives. They used the first three approaches described below.

The first approach is straightforward. It deals with evidence of direct precedence, of $A \prec B$[9]. Direct evidence means that in the training material significantly more instances of A preceding B were found. If such direct evidence is found, the decision that is supported by the evidence is made, i.e., the system predicts the ordering that contains $A \prec B$ over the one where $B \prec A$. This approach was also used by Lapata and Keller (2004) who used web-based n-grams as the underlying data.

The second approach deals with transitive evidence. In case of transitive evidence of precedence, if there exists direct evidence for $A \prec B$ and for $B \prec C$, there is transitive evidence for $A \prec C$. If such transitive evidence is found to be statistically significant, the decision is made that is supported by the evidence.

The third approach deals with clusters of pre-modifiers. When dealing with evidence for clusters of prenominal modifiers the system looks for evidence of $X \prec Y$,when $A \in X$ and $B \in Y$. Again, if such evidence is statistically significant, the decision is made that is supported by the evidence.

Malouf (2000) also presents a system for ordering prenominal modifiers. Next to the approaches by Shaw and Hatzivassiloglou (1999) as discussed above, Malouf (2000) explored four new approaches. These four approaches are discussed below.

The fourth approach describes the use of maximum likelihood estimates of bigrams of adjectives. This produces a system where there is both a likelihood for $A \prec B$ and $B \prec A$. The ordering with the highest likelihood is then chosen.

The fifth approach uses memory-based learning. In this approach a memory based classifier is trained on morphological features of sets of adjectives, with as class the adjective that is preceding. The prediction made by this classifier is followed. Vandekerckhove et al. (2011) also use the memory-based approach to model overeager abstraction for adjective ordering.

The sixth approach determines positional probabilities. This probability is calculated independently for all adjectives. The ordering that gives the highest

[9] Where $A \prec B$ stands for A precedes B.

combined, independent probability is chosen. To clarify, if there are two adjectives (A and B) then the chance of A being in the first position and B being in the last position are multiplied independently and compared to the situation when it is the other way around. So $P(\text{first}(A)) \times P(\text{last}(B))$ is compared with $P(\text{last}(A)) \times P(\text{first}(B))$.

The seventh approach uses a combination of the fifth and sixth approach. A memory-based learner is trained on both morphological and positional probability features. The classification of the learner is used as prediction.

Finally, both studies conclude that they have introduced an approach that partly solves the problem. Malouf's final result of 92% prediction accuracy is high for such a task. This result can be compared to the result achieved by a naïve approach as also reported by Malouf (2000). When employing a back-off bigram model on the first one million sentences of the British National Corpus, the model predicted the correct ordering in around 75% of the time. This leads him to conclude the following.

> ..., machine learning techniques can be applied to a different kind of linguistic problem with some success, even in the absence of syntagmatic context. (Malouf 2000, p.7)

More recently Mitchell (2009) introduced an class-based ordering of prenominal modifiers. She automatically assigns each modifier a positional-class based on its frequent positions of appearance. In the classification stage these classes are combined to determine the best-ordering of the prenominal modifiers. The performance of the system is 89.63%, comparable to the performance achieved by Malouf (2000).

Chapter 3

Experimental Setup

In this chapter we describe the experimental setup for our investigations. The setup is modular. We start by giving an overview in Section 3.1. In Section 3.2 we describe the generation of alternative sequences. The alternative sequence generator depends on the task under investigation. In Section 3.3 we discuss the selection between different alternative sequences. In Section 3.4 we explain the evaluation performed after the selection process. So finally, in Section 3.5 we give some background on suffix trees and suffix arrays. We use suffix arrays as the underlying data structure for our language-model implementations We note that the use of suffix arrays is not essential for the setup of the experiments, as it only serves as a way to count sequences efficiently. However, the use of suffix arrays has practical implications.

3.1 Flowchart of the Experiments

In this section we provide an overview of the experimental setup (see Figure 3.1). Our system consists of 13 steps (0, 1, ..., 12). We note that in this chapter the numbers between parentheses refer to the thirteen steps of the flowchart. In our flowchart the ovals refer to data, and the boxes to the actions which we perform on the data (in other words: they are the programs that we have developed or adapted). The Language-Model Environment (LME) (5, 6, 7) is the core part of the system. In the remainder of the thesis we will describe how we modify this part. In general, the steps (0, 1, 2, 3, 4) and (8, 9, 10, 11, 12) remain constant over all experiments. In some situations there are small changes (e.g., when annotations are added); they will be given explicitly in the text. Below we describe all the steps of the setup.

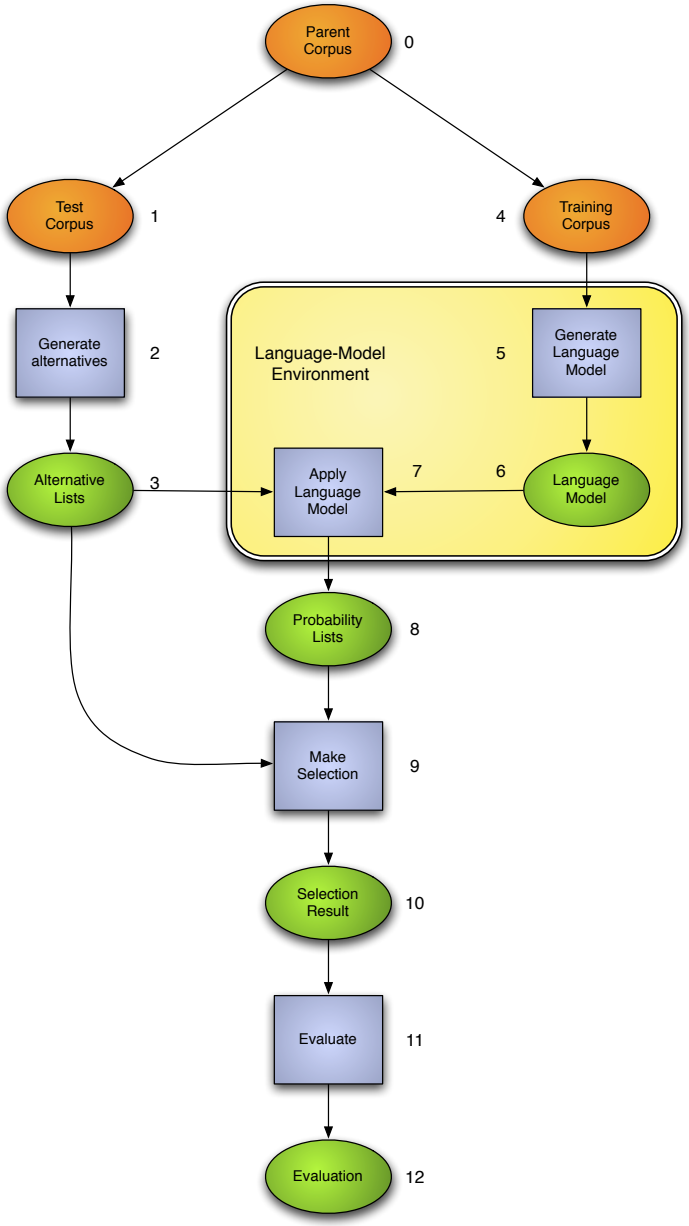


Figure 3.1: Flowchart visualizing our experimental setup. Data is represented by ovals and programs are represented by boxes. All elements of the flowchart are numbered.

Our point of departure is the parent corpus (0). In our investigation we use the British National Corpus (BNC) (see Leech et al. 1994). The BNC is a representative sample of modern-day written English. It consists of some hundred million tokens[1]. From the parent corpus, the system takes a test corpus (1) and a training corpus (4). Usually the test corpus and the training corpus are both taken from the same parent corpus in a $1 : 9$ ratio (cf. Weiss and Kulikowski 1991). We use $\frac{1}{10}$-th of the corpus for testing and the other $\frac{9}{10}$-th for training.

The training part of the corpus (4) helps to build the Language-Model Environment. The Language Model (LM) (6) is created by the *Generate Language Model* (GML) program (5). In the LME, the LM (6) forms the internal input for the *Apply Language Model* (ALM) program (7). The alternative lists (3) are the external input of the ALM (7). The output is written to (8) in the form of probability lists.

The LME will be replaced from chapter to chapter to examine the different approaches. Here we already mention the following. In Chapter 4 we replace this part by n-gram-based language models which are flexible in the size of the n-grams. In the subsequent chapters we replace the LME by n-gram-based language models to which annotations are added. In Chapter 5 the annotations are locally dependent and in Chapter 6 they are more complex.

In brief, the process starts with the test corpus (1). Then we generate (2) lists of alternatives (3). There are three different *generate alternatives* programs, one for each sequence selection problem. The three corresponding *generate alternatives* programs (2) are discussed in detail in Section 3.2. To run experiments on the different sequence selection problems we only have to change the generate alternatives program (2). Below we provide an example of the whole process for a confusible problem. For each oval in the flowchart we give an example with a reference in square brackets.

In the examples 6 and 7 we show a straightforward example of a sentence (Example 6) and the alternatives generated from that sentence (Example 7). These example alternatives are based on the confusible set $\{then, than\}$. We mark the differences in the generated alternatives by using a bold typeface. We remark that the original sentence is also generated as an alternative.

Example 6. *The pen is mightier than the sword.*

[sentence in (1)]

Example 7. *The pen is mightier* **then** *the sword.*
The pen is mightier **than** *the sword.*

[alternatives in (3)]

[1] To be precise, 111,851,659 tokens.

When the sets of alternative sequences (3) are generated we use the ALM (7) on it. The ALM assigns a probability to each of the alternatives. The outcomes are saved as a list of probabilities (8). For each set of alternatives, a separate list of probabilities is stored. For instance, a possible probability set for Example 7 is shown in Example 8. Here the first sequence has been assigned a probability of 0.3 and the second sequence a probability of 0.6.

Example 8. *The pen is mightier* **then** *the sword... 0.3*
The pen is mightier **than** *the sword... 0.6*

[probabilities in (8)]

The list of alternatives (3) and the corresponding list of probabilities (8) are used to make a selection (9). This process is discussed in more detail in Section 3.3. The *make selection* program (9) selects the most likely sequence and outputs this as a selection result (10). For each alternative sequence set this selection result consists of (a) the part of the original sequence (as contained in the corpus) that was modified and (b) the modified part of the sequence as contained in the most likely alternative sequence. Labels for the selection are detected automatically. For instance, in Example 7 the differences between the sequences are **then** and **than**; they give us two selection labels as a result (i.e., **then** and **than**). We now look back to our example of an alternative sequence set in Example 7 and to the (sample) probability list for the sequence set given in Example 8. Using these probabilities for the alternative sequences the selection made would be **than**, as shown in Example 9.

Example 9. **then** ... *0.3*
than ... *0.6* ←

[selection in (10)]

Finally, the selection result (10) is used by the *evaluation program* (11) to generate an evaluation (12). An example of such an evaluation is shown in Example 10 (the outcome is fabricated).

Our main measurement is the accuracy score of the prediction. Our reasoning for this is as follows. We are interested in the performance of the LME in terms of making a correct choice from a pre-generated set of alternatives. Hence, accuracy is the most precise measurement of this choice. We discuss the selection result (10) and evaluation (12) in more detail in Section 3.4.

Example 10. $Accuracy = 0.700$

Having described the flowchart globally by all its constituents, we are now ready to take a closer look at the programs. We will discuss generate alternatives (2), make selection (9), and evaluate (11) in Sections 3.2, 3.3, and 3.4, respectively.

3.2 Alternative Sequence Generation

For each of the three problems we have built a different alternative sequence generator. The three alternatives generators occur on position (2) in our flowchart, shown in Figure 3.1. The corresponding generator for each of the problems outputs a set of alternative sequences. In Subsections 3.2.1, 3.2.2, and 3.2.3 we describe briefly the triggers and alternative generation for each problem.

3.2.1 Confusibles

For the confusible problem we use the 21 confusible sets as introduced by Golding and Roth (1999). If a member of any of these sets is encountered it triggers the alternative generation. In other words, the trigger is the occurrence of a member of a given set of confusibles. The 21 sets and the number of times they occur in the British National Corpus are listed in Table 3.1. The alternative generation is shown in Example 11.

When generating alternative sequences for each member of the confusible set, we generate all alternatives. So, if we were to encounter the word *sight* in the running text three alternatives would be generated (see Example 11).

Example 11. *It was a lovely* **sight**.
It was a lovely **cite**.
It was a lovely **site**.

3.2.2 Verb and Noun Agreement

For the trigger of the verb and noun agreement problem we use the tagging as present in the British National Corpus. When a verb or a noun is encountered, it triggers the alternative generation. In other words, our trigger is the occurrence of a verb tag or a noun tag. If a verb tag is encountered, we use an inflection list for our generation process. An example verb inflection list for the verb *speak* is presented in Table 3.2.

Example 12. *He* **spoke** *softly.*
He **speak** *softly.*
He **speaks** *softly.*
He **speaken** *softly.*
He **spoken** *softly.*

confusible set	**# occurrences in BNC**		
accept, except	9,424	10,025	
affect, effect	4,860	22,657	
among, between	21,790	86,335	
begin, being	7,232	84,922	
cite, sight, site	288	6,352	9,612
country, county	30,962	10,667	
fewer, less	2,897	37,276	
I, me	663,660	122,477	
its, it's	144,047	114,105	
lead, led	14,223	15,468	
maybe, may be	9,541	36,539	
passed, past	10,120	25,203	
peace, piece	8,561	9,383	
principal, principle	4,781	7,832	
quiet, quite	5,969	38,836	
raise, rise	6,066	10,304	
than, then	139,531	149,237	
their, there, they're	223,820	295,546	22,466
weather, whether	5,787	32,735	
your, you're	117,864	34,242	

Table 3.1: The confusible sets as used by Golding and Roth (1999) with the number of the occurrences, for each element of the set (in order), in the British National Corpus.

Form	Example
base	*speak*
infinitive	to *speak*
third person singular	*speaks*
past	*spoke*
-*ing* participle	*speaking*
-*ed* participle	*spoken*

Table 3.2: Example of an inflection list, demonstrated by the verb *speak*.

When generating alternative sequences for each verb or noun we generate the full alternative set for the verb or noun. For a verb this means that all inflections of the verb are generated. For a noun this means that the singular and plural form are generated. For a verb it means that an inflection list is generated with the singular and plural forms of the present tense, the past tense, the present participle and gerund form (-ing), and the past participle (-ed). The set of alternatives for all verbs and nouns encountered is generated using the CELEX database as described by Baayen et al. (1993). So, if we were to encounter the word *spoke* in the running text five alternatives would be generated (see Example 12). We remark that the alternative *he speaks softly.* is also a correct sentence. However, this alternative does not match the gold-standard text. In this thesis we measure how well the alternative sequence selection system recreates the original text, so this alternative, if selected, would be counted as incorrect.

3.2.3 Prenominal Adjective Ordering

For the prenominal adjective ordering problem we again use the tagging as present in the British National Corpus. When two or more subsequent prenominal adjectives are encountered, the alternative generation is triggered. So, our trigger is the occurrence of two or more subsequent prenominal adjectives, i.e., in front of a noun. All possible orderings are generated from the set of adjectives.

In Example 13 we show a fabricated output of all alternative sequences. The number of alternatives generated for a sequence of x adjectives is $x!$[2].

Example 13. *The* **large wooden red** *cabin.*
The **large red wooden** *cabin.*
The **wooden red large** *cabin.*
The **wooden large red** *cabin.*
The **red wooden large** *cabin.*
The **red large wooden** *cabin.*

3.3 Alternative Sequence Selection

The alternative sequence selector (9), selects the sequence with the highest assigned probability. This program also assigns a label to the selection made. The inputs to the make selection program are alternative lists (3) and proba-

[2] Assuming all adjectives are unique, as is usually the case.

bility lists (8). It uses the probabilities to make a selection between alternative sequences.

The alternative sequence selector is implemented as follows: (1) the alternative lists contain all the textual sequences, and (2) the probability lists contain only the probabilities with the proper reference to the alternative lists. For readability, in the examples given above, we have replaced the reference by the full text of the sequence.

The make selection program selects the alternative sequence with the highest probability. Based on the alternative sequence selected, the corresponding selection label is automatically generated.

3.4 Alternative Sequence Evaluation

We evaluate the selection result by comparing its value to the value of the original sequence. When the values match, the prediction is counted as correct. When the values do not match, the prediction is counted as incorrect. We stress that this means that sometimes predictions of correct alternative sequences are counted as incorrect as they do not match the original input. In effect we are measuring the ability of the system to regenerate the original sequence. As evaluation measure we use the accuracy measure, i.e., the number of correct predictions divided by the total number of predictions.

For determining the significance of our predictions we use McNemar's test (McNemar 1947). Our experiments provide predictions on the same series of alternative sequence sets. Using McNemar's test we can calculate the significance of the differences between the series.

The null hypothesis of the statistical test is that both series of measurements are taken from the same distribution. The result will clarify whether or not the null hypothesis should be rejected. If we reject the null hypothesis we conclude that the difference between the two compared series of predictions is significant. Thus, McNemar's test is a χ^2 test. The formula for calculating χ^2 is given in in Equation 3.1.

$$\chi^2 = \frac{(B - C)^2}{B + C} \tag{3.1}$$

In the equation the values for B and C are given by the non-matching cells in a binary confusion matrix. We give the shape of the confusion matrix (Table 3.3). So, B and C are the counts of the elements on which both series disagree. In the case of B, it is the count of the number of times the first series had a positive

	Positive	Negative	
Positive	A	B	$A+B$
Negative	C	D	$C+D$
	$A+C$	$B+D$	N

Table 3.3: A table showing the binary confusion matrix as used for applying McNemar's test. From top to bottom we show positive and negative items from the first series. From left to right we show the matching positive and negative items from the second series.

outcome and the second series a negative outcome. In the case of C it is the other way around. For the calculations we use the R package (R Development Core Team 2010).

3.5 Data Structures Used

In this section we motivate our preference for suffix arrays. In most language models straightforward hash tables are used (cf. Stolcke 2002). A hash table is an array with {key, value} pairs where the key is translated into an index on the array by a linear-time hashing function. A suffix tree is a tree-like data structure that stores all the suffixes of a sequence as paths from the root to a leaf. A suffix arrays is an arrays of all, (lexically) sorted, suffixes of a sequence.

Below we briefly review the historical development from hash table to suffix array. The idea of using suffix trees and suffix arrays is a spin-off from Zobrist's ideas on hash tables. Zobrist (1970) used a hashing function based on the content of a game-tree[3]. In game playing, new ideas and applications of hash tables were further developed in Warnock and Wendroff (1988). They called their tables *search tables*. Parallel to this development there was ongoing work on tries and indexing. Starting with the concept of tries as described by Knuth (1973, p. 492), concepts such as suffix trees as described by Ukkonen (1995) were developed. These suffix trees in turn lead to the development of a more efficient data structure (in terms of memory use), the suffix array, as introduced by Manber and Myers (1990).

In implementations of statistical language models, suffix trees and suffix arrays can be used for the underlying data structure (cf. Yamamoto and Church 2001, Geertzen 2003). In typical models we see hash tables with stored probabilities assigned to a key, i.e., an n-gram (cf. Stolcke 2002). Hash tables (Zobrist 1970) store a single value for a key, for instance, a probability for an n-gram.

[3] In the case of Zobrist (1970) these are game-trees for the game of chess.

Suffix trees and suffix arrays provide access to counts and positions of all subsequences, i.e., any subsequence of any length of the training data. We observe that suffix trees are mainly used within bioinformatics, where they facilitate counting subsequences. In language models, suffix trees play a subordinate role, since they: (1) use more storage than a singular hash, and (2) are complex in relation to their use.

For our language-model implementations, we use suffix arrays as the underlying data structure. Suffix arrays have been proposed more recently and are strongly related to the applications of the suffix trees. For the suffix array and the suffix tree, the underlying approach to the data is rather different. As will be described below, the common property is that both data structures provide a searchable access to all suffixes of a sequence.

For a proper explanation we use the following. For hash tables we refer to the literature (Zobrist 1970, Knuth 1973, Baase and Gelder 2000, and Stolcke 2002, p. 513–558), for search tables to Warnock and Wendroff (1988), and for tries to Knuth (1973, p.492). Below, we explain suffix trees in Subsection 3.5.1, followed by suffix arrays in Subsection 3.5.2, and enhanced suffix arrays in Subsection 3.5.3.

3.5.1 Suffix Trees

A suffix tree is a tree-like data structure that stores all suffixes of the sequence. In our example, we use the word *robot*. In *robot* there are five character suffixes: $\{$*t*, *ot*, *bot*, *obot*, *robot*$\}$. A suffix is a subsequence of which the last element is also the last element of the input sequence. When displaying a suffix tree we use a trie as data structure and the alphabetical order as in Figure 3.2. The suffix tree is built in such a way that every path from the root of the tree to its leaves represents a single suffix. The leaves of the suffix tree contain an index back to the start of the suffix in the input sequence. We represent indices by numbers that start at 0. An example for the string *robot* is shown in Figure 3.3. The values in the leaves of the suffix tree point back to the index of the place in the string where the suffix starts.

Suffix trees are a well-known data structure with many applications in natural language processing (specifically string processing) and other fields such as bioinformatics. We mention three applications:

1. With suffix trees we can *search for a query sequence* in $O(m)$ time, where m is the length of the query, the results are (a) the existence of

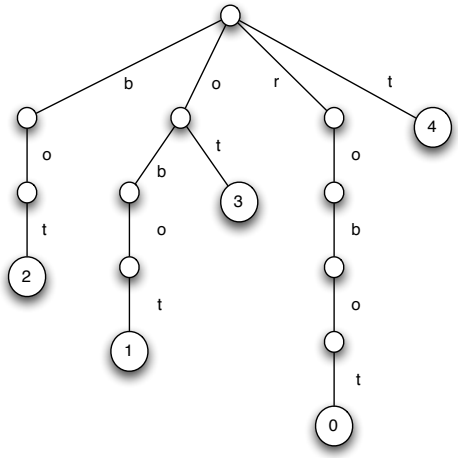


Figure 3.2: Suffix tree for *robot*.

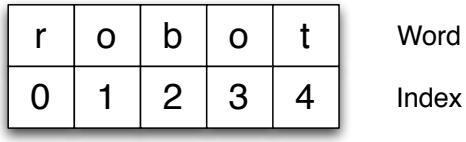


Figure 3.3: Index example on *robot*.

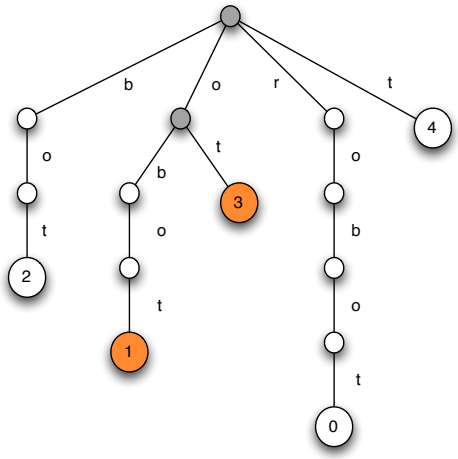


Figure 3.4: Suffix tree counting example for *robot*. The pattern searched is *o*.

the query, (b) the positions of the query in the sequence, and (c) the number of occurrences in the sequence[4].

2. We can *find the longest repeated subsequence* efficiently.
3. We can *find the shortest substrings* that occur only once.

Suffix trees can be constructed in linear time with regards to the length of the input. The state-of-the-art algorithm for the online linear-time construction of suffix trees was introduced by Ukkonen (1995). Later on, Gusfield (1997) explained this construction algorithm more clearly.

A well constructed suffix tree can be used to find the number of occurrences of substrings efficiently. To count the number of occurrences the tree is traversed from the root node following the path indicated by the query to a node in the tree. At that point we count the number of leaves under that node[5]. This is illustrated for the *robot* example with the query *o* in Figure 3.4. The number of leaves under the path for *o* is two, with suffix index 1 and 3.

There are efficient implementations available on the internet. A representative example of such an efficient implementation is by Van Zaanen (2010), who

[4] Provided this extra bit of information is stored at construction time.

[5] We note that for this application we would store this information in the nodes itself, speeding up this step significantly.

implemented Ukkonen (1995)'s algorithm for building suffix trees, in template-based C++.

3.5.2 Suffix Arrays

In the 1990s, a data structure related to suffix trees called a suffix array was developed for operations on sequential data. Manber and Myers (1990) introduced the concept of a suffix array. The data structure underlying suffix arrays deviates considerably from that of suffix trees. The development of suffix arrays is motivated by more efficient use of memory. As a side comment, we remark that storing and reading in a suffix array from disk is also more straightforward than reading in a suffix tree.

Suffix arrays are related to suffix trees in two ways: (1) a suffix tree can be converted to a suffix array in linear time, and (2) suffix arrays can support the same operations as suffix trees.

A suffix array is an ordered list of all suffixes in a sequence. The suffixes are typically ordered alphabetically. Obviously, storing all suffixes as separate sequences is rather inefficient, e.g., for storing all n suffixes of a sequence of length n it needs $\frac{1}{2} \times (n + n^2)$ memory. Therefore, only a list of indices on the original input sequence is stored, denoting the positions in which the suffixes start. In Figure 3.5, we illustrate the way of storing by the example suffix array for *robot*, including indices on the original sequence.

An interesting aspect of suffix arrays is that suffixes that share a common prefix are grouped together in the array. It means that the suffix array can be used to locate the position and number of all infixes of an input sequence. It is done by finding all suffixes that start with the given infix. Since they are grouped together, they can be found efficiently.

Suffix arrays have much lower memory requirements than suffix trees. The worst case memory usage of a suffix tree is $\Theta(m|\Sigma|)$[6] with m the length of the sequence and Σ the alphabet. In contrast, suffix arrays are $\Theta(m)$ in their memory utilisation, regardless of the alphabet $|\Sigma|$.

Building a suffix array using a regular sorting algorithm takes $\Theta(n^2 \log n)$ time, where n is the length of the input. Typical sorting algorithms use

[6] When using the notation Θ we refer to the worst-case complexity or space utilisation of an algorithm, conversely O (big O) denotes the average case complexity or space utilisation. This is analogous to the literature (Knuth 1973, Baase and Gelder 2000).

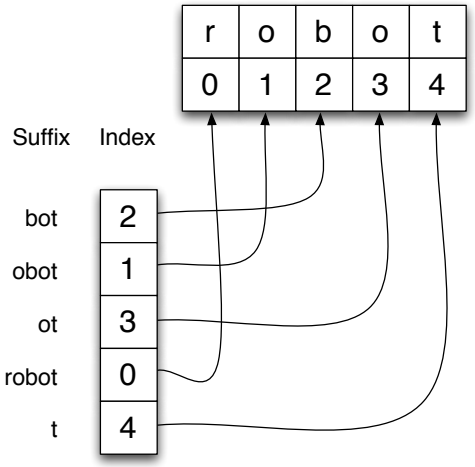


Figure 3.5: Suffix array example for *robot*. The index blocks form the suffix array. The arrows denote the pointers to the original, indexed input sequence (*robot*). Left of the suffix array we have listed the suffices.

$\Theta(n \log n)$ time to sort a sequence[7]. However the sorting of the prefixes of the suffixes can depend on multiple consecutive positions in the original input. This means that we have to perform sequence comparison which may need symbol comparisons of at most n positions. It results in a naive construction time of $\Theta(n^2 \log n)$ for a sequence of length n.

During recent years, many sorting algorithms have been developed that can construct a suffix array more efficiently with respect to time requirements. The fastest algorithms run in $O(n)$ time. They build a suffix tree first (which can be done in linear time) and then obtain the sorted suffixes by a one-pass traversal of the suffix tree. However, these algorithms need a working space of at least $15 \times n$ (see Manzini and Ferragina 2004).

We remark that Manzini and Ferragina (2004) also proposed a reasonably fast algorithm that needs a working space of only $5 \times n$. The reasonable fast algorithm works by partially sorting the array into buckets of which the suffixes start with the same x tokens and afterwards sorting each of these buckets with a blind trie. This strategy is called *deep-shallow sort*.

[7] Typical fast sorting algorithms such as mergesort have a worst case complexity of $\Theta(n \log n)$ (Baase and Gelder 2000, p. 174–177). There are many other related and relevant sorting algorithms such as bucket sort, radix sort, trie sort, etcetera. The interested reader is referred to the literature (Knuth 1973, Baase and Gelder 2000).

There are two issues we may encounter when using suffix arrays: (1) there were no free, flexible implementations available[8] at the time that we decided to use suffix arrays, and (2) searching for a (sub) sequence in a suffix array is not linear in time with regards to the pattern length, but logarithmic in time with regards to the length of the sequence on which the suffix array was built[9]. There are solutions to both issues, which we will discuss below.

Since there was no implementation available that met our needs we created our own implementation. We used as guidelines the articles by Abouelhoda et al. (2004) and Manzini and Ferragina (2004). We aimed at achieving a quite flexible data structure. So, we decided to write the data structure in template-based C++, which means that it can be applied to any sortable sequence[10]. As our implementation choices are strongly biased towards reducing the memory, we have implemented the deep-shallow sort strategy introduced by Manzini and Ferragina (2004).

After constructing the initial array of suffixes we performed a bucket sort. For the bucket-sort we employed the built-in sort algorithm of the STL library. This is a quite efficient implementation using $\Theta(n \log n)$ time (cf. Knuth 1973, Baase and Gelder 2000). As we sorted to depth x the number of sequential comparisons performed could be at most x. So, we could infer that the time complexity of this sort is at most $\Theta(x \times n \log n)$. The implementation resulted in an almost sorted list of suffixes.

To each bucket of the bucket-sorted list we then applied a blind-trie sort in turn. For the blind-trie sort we built, from the suffixes of a single bucket, a trie. This trie is then traversed in sorted order and the ordering of the suffixes is read-out. This kind of trie is also known as a *patricia tree* (Morrison 1968).

In the suffix array it takes some $O(\log n)$ time to find a pattern due to the lack of an index structure. Here we remark that the suffix tree is itself an index structure. The question is: can we combine this property of the suffix tree with the memory utilization of the suffix array somehow? The affirmative answer leads us to the concept of *enhanced suffix arrays* (see Abouelhoda et al. 2004).

---

[8] Most implementations available online pose restrictions on the size of the alphabet that is used making them not suitable for word-based n-gram models.

[9] Without an index, locating the position of a specific element of a sorted list takes $\Theta(\log n)$ steps when using binary search (Baase and Gelder 2000, p. 56-57).

[10] For a considerable sorting speedup we have eliminated the bounds check from the sorting phase. Due to this elimination the input sequence has to have a unique largest element in the last position.

3.5.3 Enhanced Suffix Arrays

Abouelhoda et al. (2004) suggested several improvements to a plain suffix array structure. Together, the suffix array and the improvements form the enhanced suffix array. The enhanced suffix array enables, amongst other things, searching in linear time. The extension consists of two parts: (1) a longest-common-prefix (lcp) value table, and (2) a child table representing the implicit suffix tree structure. Below, we discuss both in more detail. For our application, the most important enhancement is that of the *child table*.

Enhancement 1: LCP

The first enhancement is the longest-common-prefix (lcp) value. The lcp value is a positive integer that denotes the length of the common prefix of two elements. Two examples illustrate this idea. First, the lcp value of *monkeys* and *robot* is 0. Second, the lcp value for *road* and *robot* is 2 as the longest common prefix between the two is *ro*, which has length 2. This lcp value is stored for each position in the suffix array, denoting the length of the longest common prefix with the previous suffix in the array. We show an example of lcp values in an array in Figure 3.6.

Position	Index	Suffix	LCP
0	2	bot	0
1	1	obot	0
2	3	ot	1
3	0	robot	0
4	4	t	0

Figure 3.6: Suffix array longest-common-prefix example for *robot*.

We use the lcp values for the construction of a virtual suffix tree. For this purpose we use the concept of *lcp intervals*. An lcp interval defines the interval that corresponds to the range of suffixes (in the suffix array) with a specific prefix. In Figure 3.6 the lcp value 1 indicates that the given position shares a prefix of length 1 with the previous item. Moreover, the 1 is preceded and succeeded by lower values. Since the lcp value indicates the relation between the

current element of the array and the preceding one, we may establish the length of the lcp interval as 2. An interval $[i \ldots j], 0 \leq i < j \leq (n-1)$ with n the length of the sequence, is an lcp interval of the lcp value l if the following four conditions hold:

1. $\text{lcp}[i] < l$
2. $\text{lcp}[k] \geq l$ for all k with $i+1 \leq k \leq j$
3. $\text{lcp}[k] = l$ for at least one k with $i+1 \leq k \leq j$
4. $\text{lcp}[j+1] < l$.

Below we explain the conditions.

Ad 1 Since the lcp value is l, the $\text{lcp}[i]$ should be lower than l. Otherwise element $i-1$ would also belong to the lcp interval.

Ad 2 All elements within the lcp interval must at least share a prefix of size l.

Ad 3 There needs to be at least one element in the interval that has an lcp value of l. Otherwise, the shared prefix of the interval would have a length greater than l.

Ad 4 The element directly after the lcp interval does not share the prefix of length l with the lcp interval, otherwise it would also belong to the lcp interval.

The lcp intervals can have smaller lcp intervals embedded within them, recursively. These recursive intervals have a tree structure and are called an lcp interval tree. The lcp interval tree is implicit and has the same structure as the suffix tree. We show the (recursive) lcp intervals of our example *robot* in Figure 3.7. The root lcp interval encompasses the entire suffix array. This interval constitutes the lcp interval with lcp value 0. The root interval branches to depth 1 into four lcp intervals, viz. {*bot*,*o*,*robot*,*t*}. Of these, only the lcp interval for *o* has more elements than 1, namely 2. Therefore, this specific interval branches once more into {*bot*, *t*}.The whole structure is given in Figure 3.8.

Enhancement 2: Child Table

Using the lcp values all the lcp intervals can be computed in a single pass (with a stack-based algorithm) over the suffix array. However, the resulting representation does not allow us to search for a desired internal node (i.e., a desired pattern) efficiently. To improve the efficiency of the search we use

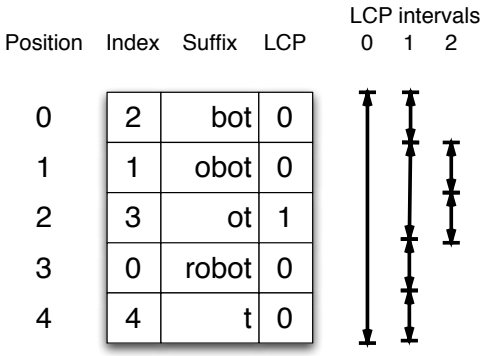


Figure 3.7: Suffix array lcp intervals example for *robot*.

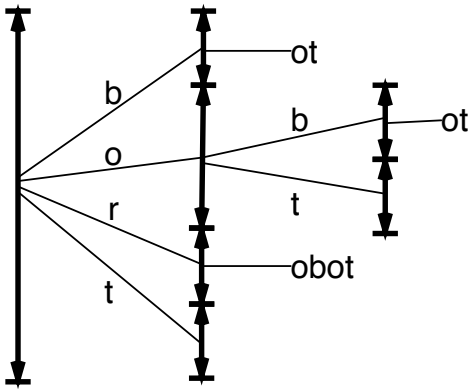


Figure 3.8: Lcp tree example for *robot*. The lcp intervals can be also seen in Figure 3.7.

Position	Index	LCP	ChildTab Down	ChildTab Next	Suffix
0	2	0	-	1	bot
1	1	0	2	3	obot
2	3	1	-	-	ot
3	0	0	-	4	robot
4	4	0	-	-	t

Figure 3.9: Enhanced suffix array example for *robot*.

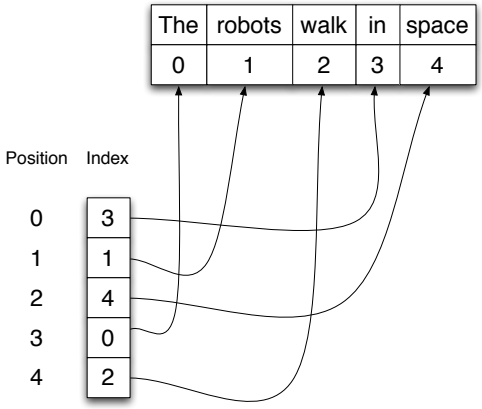


Figure 3.10: Suffix array example for *The robots walk in space*.

the childtable representation. The addition of a child table makes the implicit suffixtree structure more explicit, so we can search for patterns in $\Theta(n)$ time with regards to the pattern length.

In order to access the implicit suffixtree structure efficiently we store the jumps through the suffix array that we need for top-down traversal of the implicit suffix tree in an additional support array. The idea is that for each interval we can easily determine the children of that interval (which are also intervals themselves), and that we can do so recursively. Therefore, we need indices for the next element of the current depth of the tree. We also need an index for the first element of the next lcp interval one level deeper. Of course, the interval $[0 \ldots n-1]$ is always a valid lcp interval, giving us a start position to work from.

An example of a regular suffix array for our example *robot* enhanced with the lcp values and the child table is given in Figure 3.9. We have already shown the corresponding lcp interval tree in Figure 3.7 and Figure 3.8. Below we explain the contents of *next* and *down*. Next always points to the next branch at the same depth. For depth 1, we have four intervals (see Figure 3.7). For each first element of each of the intervals the next value points to the first element of the next interval. So, we find the values 1, 3, and 4. The down value points to the first element of the second branch one level lower. Finally, we remark that in our research only top-down tree traversal is applied. For tasks that require bottom-up tree traversal approaches, Abouelhoda et al. (2004) have introduced a third index, called *up*. The use of the up index falls beyond the scope of our research.

The construction of the child table can be done, analogously to the lcp value array, in a single pass over the suffix array. However, building the child table depends on the presence of the lcp table. Therefore, two passes over the suffix array are needed to fill the support structures that allow access to the implicit suffixtree structure after regular construction.

From Letters to Words

We created an efficient, template based, C++ implementation of these techniques, which is used throughout this thesis. We exploit this implementation to answer queries for n-grams of words.

It is possible to construct a suffix array based on sequences of words instead of sequences of characters. In doing so we gain the ability to answer word-based n-gram queries more efficiently. We give an example of a suffix array built on the short sentence *The robots walk in space*. The sentence contains five

suffixes: (1) *space*, (2) *in space*, (3) *walk in space*, (4) *robots walk in space*, and (5) *The robots walk in space*. We show the suffix array in Figure 3.10.

Having described of the enhanced suffix arrays, we are ready to perform our experiments. In Chapter 4, 5, and 6 we describe (1) the LME as applicable to the specific situation at hand, i.e., without annotations, with part-of-speech annotations, and with dependency annotation, and (2) the results achieved using that LME.

Chapter 4

Models without Annotation

This chapter describes a language-model environment (LME) that is to be characterised as a model without added annotation on top of the words. Using this straightforward LME and the corresponding LM, we investigate the flexibility of the n-gram. This means that, we explicitly choose not to set the size of the n-gram in advance. However, we do the comparison with fixed size n-grams in the preliminary experiments. They are discussed in Section 4.2. In this chapter, we examine RQ1 and RQ2 with respect to this model for the three alternative sequence selection problems. Below we reiterate both RQs.

> **Research Question 1:** Is there a need to predetermine or limit the size of the n-grams used in language models? Is there an inherent advantage or disadvantage to using a fixed-size n?

> **Research Question 2:** If the size of the n-grams is not fixed in advance, how can we still generate comparable distributions when we select among alternative sequences?

The course of this chapter reads as follows. In Section 4.1 we describe n-gram models and several of their aspects. We then discuss (1) the n-gram size, (2) smoothing, (3) interpolation, and (4) back-off.

In Section 4.2 we report on experiments to determine variables related to the four items discussed in Section 4.1. The experiments are performed on the confusibles disambiguation task and lead to initial conclusions. These conclusions have guided the precise design of the full range of our experiments. Based on these experiments we introduce a new back-off method for alternative sequence selection.

In Section 4.3 we complement the language-model environment as described in Chapter 3 with a precise description of the back-off. The section is brief and only provides essential details about the experimental setup.

In Section 4.4 we describe the results of our experiments on the three alternative sequence selection problems. In addition to the different problems, we investigate the influence of the flexibility in terms of the n-gram size. The results of the experiments are analysed and related to RQ1 and RQ2.

In Section 4.5 we explicitly formulate our answers to RQ1 and RQ2 for the models without annotation. In Section 4.6 we provide a summary and our conclusions.

4.1 Basics of n-gram Language Models

Statistical language models (SLMs) based on n-grams assign probabilities to sequences from a language[1]. Jelinek (1998) gives a detailed overview of these models in his book. The advantage of using n-grams is that they describe the sequences in a language in a compact representation. The main features of the models are the simplicity of calculating the probabilities for sequences, and their speed.

Language modelling deals with (1) determining the likelihood of a sequence (or part of a sequence) and (2) determining the validity of a sequence (or part of a sequence). In our research we focus on determining the likelihood of a sequence. Since we consider alternatives for given sentences, we do not need the concept of validity of a sequence to determine the most fitting candidate sequence. We mention that in this thesis the phrase 'determining the likelihood of a sequence' is equivalent to 'assigning a probability to a sequence'. We denote the probability of a sequence of n words, as assigned by the language model (LM), as follows.

$$P_{LM}(w_1, \ldots, w_n) \tag{4.1}$$

Equation 4.1 represents the ideal probability computed by the ideal language model. However, we do not possess a perfect model of any natural language. Instead, we approximate the probability of the sequence. Even when using a large

[1] Statistical language models are a type of grammatical language models. Roughly speaking, we may make the following distinctions. Language models deal with the *distribution* of language, grammatical models deal with the *structure* of language. However, the structure of language can be expressing in n-grams, which is the basis for k-testable machines. In the literature, Adriaans and Van Zaanen (2004), and de la Higuera (2005, 2010) give an extensive overview of the field of grammatical inference, including k-testable machines.

amount of training material, it is typical that not all the possible sequences in the language are explicitly known[2]. For instance, assume that we are computing the occurrences of complete sentences, then it will turn out that most of those sentences are not known[3] to the model. This is the sparseness problem described in Section 1.1. Later in this section, we describe three methods for dealing with the sparseness problem: smoothing (see Subsection 4.1.1), interpolation (see Subsection 4.1.2), and back-off (see Subsection 4.1.3).

The Relation between Sparseness and n-gram Size

The relation between the sparseness problem and the n-gram size is an important aspect in our research. One way to deal with sparseness is to model the probability of complete sequences from the probabilities of smaller sequences. A well-known approach is to use n-grams. They can be considered as Markov models[4] as described by Jurafsky and Martin (2000, Ch. 6). These models take into account the probability that a word occurs in the context of the previous $n-1$ words.

Taking the product of the probabilities of all words in the sequence, with $n-1$ words as context, gives us a prediction of the probability of the sequence. We express this in the Equation 4.2.

$$P_n(w_1, \ldots, w_m) = \prod_{i=1}^{m} P_n(w_i | w_{i-(n-1)}, \ldots, w_{i-1}) \tag{4.2}$$

We provide a clear example of using n-grams, specifically, of using a 3-gram to calculate the probability of the word *dog* given a sentential context. Instead of computing the probability $P_{LM}(\text{dog}|\text{The quick brown fox jumps over the lazy})$ we compute the probability $P_3(\text{dog}|\text{the lazy})$. The difference between the two probabilities is as follows. The first probability ranges over the entire sequence and the second one is restricted to the two preceding words in its immediate context. In SLMs based on n-grams, the assumption is that the second probability is an adequate approximation of the first one. The approximation is shown formally in Equation 4.3 where the probability for the word under consideration (w)

[2] With infinite languages this is always the case, with finite languages this is not the case only if the entire language is provided as training material.

[3] It assumes that we are computing occurrences of natural language. There are many classes of language, of which some contain finite languages. For finite languages it is possible to have observed all possible sequences.

[4] An English translation of one of Markov's lectures (republished in Markov 2006) recently became available, explaining the concept in Markov's own words.

depends solely on its occurrence compared to the previous $n-1$ words (wherein $w_{i-(n-1)}$ denotes the word $n-1$ positions to the left of w_i).

$$P_n(w_i|w_{i-(n-1)},\ldots,w_{i-1}) \tag{4.3}$$

All n-gram-based models make predictions about the probability of a word in a sequence based on local information bounded by n, where n is usually small (2 to 5).

An approximation of the probabilities can be extracted from the counts of occurrences of n-grams in a corpus. In the uncomplicated case, probabilities are computed by taking the relative occurrence count of the n words in sequence. We can use such models to predict the most likely word w_i following a sequence of $n-1$ words.

When dealing with natural language, the use of n-grams with large n tends to sparseness. Often the training data does not contain occurrences of the particular sequence of n symbols, even though the sequence in question is correct. Alternatively, the training data might contain too few instances of the particular sequence to estimate the probability reliably. In the first case, the probability extracted from the training data will be zero, even though the correct probability should be non-zero (albeit smaller than the probability for observed sequences). In the second case, the probability extracted will be unreliable. Specifically, Good (1953) already implied that observations with low counts may give rise to a considerable overestimation of the probability. For clarity, we remark that a correct or valid sequence should be assigned a probability greater than zero, and a sequence perceived to be incorrect by native speakers a probability of zero.

We now illustrate the problem of sparseness by looking at the number of n-grams of different sizes. Consider, for instance, the Wall Street Journal part of the Penn Treebank (Marcus et al. 1993). It consists of 1,174,170 tokens. Of these 1 million words there are around 50 thousand distinct unigrams, 400 thousand distinct bigrams, 800 thousand distinct trigrams, 900 thousand distinct 4-grams, and also 900 thousand distinct 5-grams. We show these values in Table 4.1. It can be observed that the growth in the number of distinct n-grams for size n compared to size $n-1$ sharply diminishes, and reaches 1.0 by $n=10$. A telling example can be observed in the transition from unigrams to bigrams. In the WSJ we observe the word *chapter* 77 times. However, the short sequence *the chapter* is never observed, even though this is undoubtedly a valid sequence. We note that a large part of the potential bigram space ($50,000^2$ words) consists of invalid combinations. However, the amount of valid combinations should clearly be much larger than $397,057$. Almost all 4-grams and 5-grams occur only once. There are many valid n-grams that we do not en-

n-gram	# distinct n-grams	factor
1-gram	49,152	
2-gram	397,057	8.07
3-gram	787,931	1.98
4-gram	957,812	1.22
5-gram	1,011,748	1.06
6-gram	1,029,442	1.02
7-gram	1,036,625	1.01

Table 4.1: The size of the n-gram and the number of distinct n-grams of that size. The factor is calculated by $\frac{|n|}{|n-1|}$ and denotes the multiplication factor for the number of distinct n-grams of length n, compared to distinct n-grams of length $n-1$.

counter in the corpus, and therefore not in the training material. This clearly illustrates the problem of sparseness.

A large n-gram size makes the sparseness problem worse as the number of observations in relation to the amount of valid language sequences of length n decreases strongly. However, it is nevertheless desirable to use an n-gram as large as possible. The larger the accurate n-gram, the better it is able to estimate the probability of a sequence of any length. For instance, when using unigrams, n-grams with $n = 1$, no context words are taken into account when calculating the probability. Consequently, in the sequence *The quick brown fox* is assigned a lower probability than the sequence *the the the the*[5]. This is undesirable. If we increase the size of the n-gram, effectively increasing the size of the local context, we reduce this problem until at $n = m$ (m being the length of the sequence) the context problem for computing the probabilities disappears as then $p_m(w_1 \dots w_m) = P_{LM}(w_1 \dots w_m)$, i.e., when approaching the ideal model. The larger the size of the n-gram, the more accurately the language model is able to assign a probability to a given sequence. In this scenario we assume that there are sufficient observations of the sequence in order to make a reliable probability estimation. In practice, this is rarely the case. We therefore prefer to use a large size n-gram that is at the same time not so large as to cause too much sparseness.

[5] The word *the* is one of the most frequent words in the English language, therefore in an unigram model it has a quite high probability. In the aforementioned Wall-Street Journal corpus it occurs around 33 thousand times, or around one in every 50 words.

Approaches to Reduce the Sparseness Problem

To reduce the sparseness problem we may use (1) smoothing, (2) interpolation, and (3) back-off. We remark that the boundaries between these three categories are vague and that many techniques can be seen to fit in more than one of them. Chen and Goodman (1996) provide an overview of the three approaches in many varieties. Smoothing methods redistribute the probabilities to take into account previously unseen word sequences. For interpolation methods, distributions of different order n-grams are mixed. For back-off methods, the probabilities of lower-order n-grams are used when there is insufficient data for the higher-order models. In the remainder of this section we will discuss the three classes of methods.

4.1.1 Smoothing

The first class of methods to deal with the sparseness problem we discuss is smoothing. Like all redistribution methods, smoothing methods assume an imperfect sampling of data. The methods redistribute the probabilities of the n-grams, theoretically resulting in a distribution that fits the reality better than the sample. Due to this redistribution a part of the new probabilities can be considered to be assigned to unseen events. Part of the probability redistribution is caused by a shift from more frequent n-grams to less frequent n-grams in the assigned sample. This results in all n-grams having a probability higher than zero in the model. The smoothing method originates from biology. In this situation a class is an observation of a certain type of animal, a common subdivision is by species[6]. In biology, all potential classes are given a probability greater than zero (cf. Good 1953). The consequence for SLMs is that when this type of smoothing is used even invalid sequences are now assigned a probability greater than zero. Below we discuss two specific smoothing methods out of the class of methods, viz. add-one smoothing and Good-Turing smoothing.

Add-one smoothing is one of the earliest smoothing methods. The counts of all the observations made during sampling are modified by adding one to them. So, the count of an hitherto unseen element is one. This ensures that there is always a non-zero probability assigned to a sequence. However, as clearly explained by Gale and Church (1994) the add-one smoothing method is not a good smoothing method in terms of the redistribution of probabilities. They show that the add-one smoothing method can lead to errors that are several magnitudes worse compared to real data using a test set. Additionally they state the following.

[6] It is assumed that for observing counts all potentially observed species in an area are known. Depending on the location is more or less true.

r	N_r	r^*
0	74,671,100,000	0.0000270
1	2,018,046	0.446
2	**449,721**	**1.26**
3	177,933	2.24
4	195,668	3.24
5	68,379	4.22
6	48,190	5.19
7	35,709	6.21
8	27,710	7.24
9	22,280	8.25

Table 4.2: An illustration of the effects of Good-Turing smoothing. From Jurafsky and Martin (2000). The counts N_r are bigram frequencies of frequencies taken from 22 million Associated-Press bigrams. The third line is boldfaced as it is used in a clarifying example in the text.

> [F]or Add-One to produce reasonable estimates, it is necessary that the ratio of unseen types to observed types and the ratio of all types to the training sample size be equal. There is no reason to assume such a relation between these observations. (Gale and Church 1994, p. 6)

Good (1953) wrote a seminal work on smoothing. He introduced the smoothing method now known as Good-Turing smoothing. It is one of the many smoothing methods included in Chen and Goodman (1998). It has also been extensively explained in many textbooks, e.g., Jurafsky and Martin (2000). In brief, the smoothing method is described by the following equation:

$$r^* = (r+1)\frac{N_{(r+1)}}{N_r} \tag{4.4}$$

In Equation 4.4 r is an observed count of a distinct n-gram and r^* is its revised count. N_r denotes the number of distinct tokens that have been observed with count r, this is known as the count of counts. $N_0 \neq 0$ (N_0 consists of unseen n-grams). Good (1953) estimate the number of unobserved tokens using the size of the vocabulary (please remark that this is quite arbitrary). For unigrams we use the size of the vocabulary; for bigrams the square of the size of the vocabulary. For more complex models we estimate N_0 analogously. We illustrate applying Good-Turing smoothing by an example, shown in Table 4.2. We explain the table by running through the revised count r^* for the n-grams that

occur twice, i.e., for $r = 2$. In the example $N_2 = 449,721$ and $N_3 = 177,933$, i.e., N_r and N_{r+1}. Using these values and Equation 4.4 we calculate the new, smoothed, value r^*. Consequently, we obtain $r^* = 3 \times \frac{177,933}{449,721} \simeq 1.26$. So, the new counts for n-grams of which the count was previously 2 will now be 1.26.

For further reading we refer to Chen and Goodman (1998). They provide an excellent overview of several smoothing methods. They also give a comparison. The smoothing methods mentioned include amongst others: additive smoothing, Good-Turing smoothing, Jelinek-Mercer smoothing, Witten-Bell smoothing, and absolute discounting.

4.1.2 Interpolation

Interpolation is the second class of redistribution methods to deal with the sparseness problem. Interpolation combines probability distributions over several different models into one. The idea is that a sparse, precise, and complex model is supported by less sparse, less precise, and less complex models. For instance, it is common to interpolate the probabilities of an SLM (of n grams) by one or more SLMs of n'-grams with $n' < n$. The probabilities are combined regardless of the probability of the higher-order models[7]. We briefly discuss two interpolation methods below.

The first method is *linear interpolation*. The probabilities of different SLMs are combined equally. This is a straightforward interpolation of models and suffers from many drawbacks. The most prevailing drawback is that the contribution of the lower-order model is equal to the contribution of the higher-order model, whereas we may expect in advance that the higher-order model is more precise. However, the unigram model contains fewer unique n-grams than the higher-order model (e.g., the trigram model), which implies that overall, the expected probabilities of the unigram model are higher, and so, their contribution will be larger (which is the reverse of what we aim for). That is, the probability space has to be divided over more distinct n-grams.

The second method is *weighted interpolation*. This is also a linear interpolation method.‘ Here, we also combine the probabilities of the SLMs involved. The difference between linear interpolation and weighted interpolation is that in the case of weighted interpolation the probabilities are weighted by some factor. Typically, higher-order models have a higher weight so as to compensate for the smaller amount of the probability space that a sequence occupies compared to the lower-order models.

---

[7] n-gram models are considered to be of higher-order than $n-1$-gram models.

We give an example. Assume that we have the sequence $w_1\ w_2\ w_3$ and are employing a trigram language model. Moreover, we assume that the sequence has been enhanced by start-symbols so that the immediately preceding context is defined (cf. Stolcke 2002). We then combine the probabilities as follows.

$$\begin{aligned} P_{LM}(w_i \dots w_m) &= \alpha \times \prod_{i=1}^{m} \Big(P_{LM3}(w_i | w_{i-2} w_{i-1}) \Big) \\ &+ \beta \times \prod_{i=1}^{m} \Big(P_{LM2}(w_i | w_{i-1}) \Big) \\ &+ \gamma \times \prod_{i=1}^{m} \Big(P_{LM1}(w_i) \Big) \end{aligned} \tag{4.5}$$

In the case of linear interpolation we take as the interpolation values $\alpha = \beta = \gamma = \frac{1}{|LM|}$, where $|LM|$ represents the number of different language models that are interpolated. For weighted interpolation we may use unequal values, that add up to 1 so that the sum is also a probability, to weigh the SLMs involved. In the overview article of Chen and Goodman (1998) interpolation methods are also discussed.

4.1.3 Back-off

Back-off is a third class of redistribution methods to deal with the sparseness problem. The essence of the back-off method is that it starts using less sparse models as soon as the SLM is effected by sparseness too much. So, lower-order models are used to approximate the higher-order models in cases where the probability of the higher-order model is zero (theoretically, is smaller than a chance epsilon, i.e., ϵ for cases where smoothing is also employed).

The main difference between interpolation and back-off is that for back-off only one language model is used at a time to determine the probability for a word w in the sequence; for interpolation all available language models are combined to determine the probability for a word w in the sequence. We remark that the difference is not always so explicit as stated above. There are back-off methods which use ideas from interpolation and vice versa (cf. deleted interpolation, absolute discounting). Below we discuss two specific back-off methods: continuous back-off and Katz. For explanations of several other back-off methods, we refer to Chen and Goodman (1998).

A straightforward back-off strategy is *continuous back-off*. The *continuous back-off* method uses probabilities from a single model at each position of the sequence, and prefers to use the higher-order model probabilities over the lower-order model probabilities. Such a back-off model thus provides a step-

$$w_1 w_2 w_3$$

$$\begin{array}{c} P_3(w_3|w_1 w_2) \leq \epsilon? \\ \Downarrow \\ P_2(w_3|w_2) \leq \epsilon? \\ \Downarrow \\ P_1(w_3) \end{array}$$

Figure 4.1: The illustration of the working of continuous back-off.

wise back-off. This method conforms to the Katz (1987) back-off method without a weighting factor (see also below). We illustrate the *continuous back-off* method in Figure 4.1. One sequence is given $w = w_1 w_2 w_3$.We examine the back-off method for position 3, i.e., determine the probability of w_3. Since we start with a trigram model there are two possible back-off steps, viz. (1) from trigram to bigram, and (2) from bigram to unigram. We remark that the $P_1(w_3)$ always has an answer (> 0 assuming that all words are known).

Katz (1987) introduced the so-called Katz back-off method. This is a method applicable to n-gram SLMs. In the case where an n-gram was not seen in the training set, one backs off to the $(n-1)$-gram. The probability of the $(n-1)$-gram is modified by a factor α which is estimated based on the fraction of the probability mass available for the not-occurring n-grams after smoothing (for instance, after Good-Turing smoothing; this is the proportion of the probability mass assigned to N_0). The method uses Equation 4.6 to compute P_{bo} (P_{bo} is the probability assigned by the back-off model).

$$P_{bo}(w|w_-) = \begin{cases} d_{w_-} \times \frac{C(w_- \ldots w)}{C(w_-)} & \text{if } C(w_- \ldots w) > k \\ \alpha_{w_-} \times P_{bo}(w|w_-) & \text{otherwise} \end{cases} \tag{4.6}$$

With $w_- = w_{-(n-1)} \ldots w_{-1}$. Here $C(x)$ is the count of x; α the back-off weight, d the smoothing factor, and k the back-off constant. The α and d are typically determined using a smoothing method such as Good-Turing smoothing, as is also done by Katz (1987).

Synchronous Back-off

In the case of alternative sequence selection the back-off methods discussed so far have the problem of possibly comparing probabilities of different SLMs. We

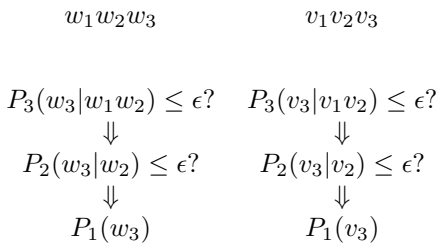


Figure 4.2: The illustration of the working of continuous back-off. In this case we show two (parallel) alternative sequences. The back-off of the two sequences is separate.

illustrate this problem with the aid of Figure 4.2. This figure shows the usage of the *continuous back-off* method discussed above for the alternative sequence selection case. Observe that the back-off steps on the sequences $w_1w_2w_3$ and $v_1v_2v_3$ are independent. So, it may occur that probabilities from different order models are compared. Comparing the values of different order models is not desirable. In general lower order models have a larger mass per unique n-gram and thus a better chance to win, whereas we prefer the higher order model as it comes closer to the ideal model.

To resolve this issue we developed a novel back-off method, called *synchronous back-off* (Stehouwer and Van Zaanen 2009a,b, 2010c). The difference with the continuous back-off method is as follows. In the continuous back-off method, a position in two similar sequences may be computed using probabilities of SLMs of different order (see the example in Figure 4.2). In the synchronous back-off method probabilities of the same SLM must be used at the same position for all alternative sequences. In the approximation, the highest-order model is preferred, i.e., the one that has at least one non-zero probability (i.e., $\geq \epsilon$) on one of the sequences at the position in question. We call the highest-order model, when combined with an unrestricted n-gram sizes the ∞-gram.

For clarity, we provide an example in Figure 4.3. The difference with Figure 4.1 is that in order to have assigned probabilities to w_3 and v_3 the reduction to a lower-order SLM may only happen synchronously (in the figure this is indicated by the word 'and'). We look at trigrams on the focus position, i.e., $w_1w_2w_3$ and $v_1v_2v_3$. If the trigram model has a non-zero probability on the first sequence and a probability $< \epsilon$ on the second sequence, the synchronous back-off method will assign the probability of the trigram model to both sequences. This will result in the second sequence being assigned an probability $< \epsilon$. When both have a probability $< \epsilon$, a back-off to a lower-order model is performed synchronously. This is in line with the idea that if a probability is

$$
\begin{array}{ccc}
w_1 w_2 w_3 & & v_1 v_2 v_3 \\
\\
P_3(w_3|w_1 w_2) \leq \epsilon? & \text{and} & P_3(v_3|v_1 v_2) \leq \epsilon? \\
\Downarrow & & \Downarrow \\
P_2(w_3|w_2) \leq \epsilon? & \text{and} & P_2(v_3|v_2) \leq \epsilon? \\
\Downarrow & & \Downarrow \\
P_1(w_3) & & P_1(v_3)
\end{array}
$$

Figure 4.3: The figure illustrates the working of synchronous back-off. In this case with two alternative sequences. The back-off of the two sequences is linked. Only if both probabilities at a higher-order model are smaller or equal than ϵ does the back-off to a lower-order occur.

$< \epsilon$ and the training data sufficient, that the sequence is less likely. We assume we observe valid language sequences much more often than invalid sequences, i.e., when one of the alternative sequences has not been observed we assume it to be less valid than the observed sequences. In short, it means that we have seen evidence supporting at least one of the alternative sequences. So, all subsequences will be assigned a probability by the same SLM, as the same position. This method is applicable for sets of alternative sequences for any size, i.e., also if there are many alternatives.

However, there is an issue with the synchronous back-off method that one has to take into account. The method only works when comparing different alternative sequences of the same length. One situation where synchronous back-off runs into problems is when the alternatives in the confusible set (in particular when occurring on the focus position) contain a different number of tokens. For example, consider the case of *your* (one token) versus *you 're* (two tokens)[8] . Also consider *onto* versus *on to*. The problem here is that for the first alternative, there is one token as *focus position*, whereas the other alternative has a two token focus position.

We wish to note that using synchronous back-off can result in a distribution which does not represent a proper probability. This means, it does not add up to one. The reason is the dynamic variation of the n-gram size. Synchronous back-off does ensure that the resulting scores[9] are comparable. In this thesis

[8] We remark that there is a linguistic reason to consider *you 're* to be two tokens. The two parts perform different linguistics functions in a sentence and are generally assigned a separate part-of-speech tag.

[9] We use the word score here instead of probability as, technically, it is no longer a probability.

we investigate making a selection between alternative sequences, having comparable scores is sufficient for that.

4.2 Towards Flexible SLMs

In this section we describe eighteen preliminary experiments and their results (partly published in Stehouwer and Van Zaanen 2009a). They are considered to have an explorative nature. We hoped that their results give us insights into the best way of performing experiments. However, we were pleasantly surprised by the results when they showed us the strength of our novel alternative sequence selection with statistical language model method, called synchronous back-off. Even early on, it was clear to us that the synchronous back-of was an approach superior to the typical statistical language approach, at least for alternative sequence selection.

The preliminary experiments fall outside our experimental setup as described in Chapter 3.

In Subsection 4.2.1 we will describe the experimental setup used in these preliminary experiments in detail. The results of the experiments are given together with conclusions in Subsection 4.2.2. As mentioned above the experiments have shaped our work. In Subsection 4.2.3 we describe their impact on our work.

4.2.1 Preliminaries: Experimental Setup

For the preliminary experiments a rather ad-hoc experimental setup was used. This enabled us to obtain quick results for a better definition of the area under investigation. The experiments were run on both Dutch and English data.

For the Dutch experiments we used the D-Coi data (Oostdijk et al. 2008) (the Dutch language Corpus initiative) as parsed by the Alpino parser. The D-Coi corpus consists of data from several different domains, such as Wikipedia, books, brochures, newspapers, and websites. The Alpino parser is described in detail by Van der Beek et al. (2001). The main results of the parser are in the form of treebanks. At the time of writing several treebanks, as generated by this parser, are available on the web at http://www.let.rug.nl/˜vannoord/trees/.

For the English experiments we used both the Reuters RCV1 corpus and the Wall Street Journal (WSJ) part of Penn Treebank (PTB). The Reuters corpus is described by Lewis et al. (2004). The PTB corpus is described in detail by

Marcus et al. (1993). In our early experiments we have used the Reuters corpus as training material. The WSJ part of the PTB was used as test corpus.

For both sets of experiments hand-picked sets of confusibles were used. For the Dutch experiments we used {*word, wordt*} (become first person and second person singular present tense), {*de, het*} (*the* non-neuter and neuter forms of *the*), {*kun, kan*} (*can* second person singular present tense, and first, second and third person singular present tense), and {*hun, hen, zij*} (*them, their/them, they*). For the English experiments we used {*accept, except*}, {*affect, effect*}, {*extent, extend*}, and {*then, than*}.

For the set of experiments on Dutch we only used trigrams. Six experimental language models were employed: (1) linear interpolation, (2) weighted interpolation[10], (3) continuous back-off, (4) synchronous back-off, (5) add-1 smoothing, and (6) Good-Turing smoothing. All six experimental language models are based on 3-grams, i.e., in case of back-off the largest model is a trigram model. Based on the results of these experiments we decided (a) not to focus on interpolation[11], and (b) to expand the size of the n-grams used. For English, we describe three types of experiments, viz. 3-gram, 4-gram, and 5-gram experiments.

So, for the set of experiments on English four experimental language models were employed: (1) continuous back-off, (2) synchronous back-off, (3) add-1 smoothing, and (4) Good-Turing smoothing. These four language models were run using 3-gram, 4-gram, and 5-gram SLMs in turn.

For these sets of experiments we used a hash-based approach to SLMs. It means that the counts and probabilities of the n-grams were stored as values attached to a key. In these cases such a key would be an n-gram.

4.2.2 Preliminaries: Results and Conclusions

In the experiments, predictions are made on sets of alternative sequences. Each alternative sequence set is characterised by one of the confusible sequence sets given above. The results were calculated as accuracy of the prediction in percentages. We remark that, if the system is not able to make a choice, e.g., due to sparseness, the specific instance is counted as incorrect.

The results are listed in Tables 4.3 (Dutch) and 4.4 (English). We note that both tables may give rise to extensive observations leading to a plethora of small findings. However, we prefer to keep the main line of our research as given

[10] We remark that the weights were set using iterative hill-climbing on a holdout set. This results in unigrams with weight 1, bigrams with weight 138, and trigrams weight 437.

[11] This decision is the reason that (1) and (2) were not used in the English experiments.

	{word, wordt}	{ de, het}	{kun, kan}	{hun, hen, zij}
Uniform linear	99.38	80.70	94.52	49.67
Weighted linear	99.38	80.91	96.42	50.76
Continuous back-off	85.24	46.50	74.39	48.59
Synchronous back-off	**99.63**	**89.91**	**97.09**	**57.95**
Add-1 smoothing	92.51	85.58	90.65	52.41
Good-Turing smoothing	92.52	85.54	90.67	52.57

Table 4.3: The table illustrates the performance achieved by the preliminary experiments on Dutch. We show experiments on the four sets of confusibles for the following trigram SLMs: (1) uniform interpolation, (2) weighted interpolation, (3) continuous back-off, (4) synchronous back-off, (5) add-1 smoothing, and (6) Good-Turing smoothing. Results are shown by their accuracy (%).

	{accept, except}	{affect, effect}	{extent, extend}	{then, than}
3-gram cont	74.12	83.40	87	78.48
3-gram sync	**88.11**	**93.78**	**95**	**94.5**
3-gram add-1	74.83	87.14	**95**	87.02
3-gram gt	74.83	87.14	**95**	87.02
4-gram cont	62.24	72.20	72	69.20
4-gram sync	**88.11**	**93.36**	**95**	**94.59**
4-gram add-1	41.26	64.73	80	64.85
4-gram gt	41.26	64.73	80	64.81
5-gram cont	62.24	68.63	70	66.44
5-gram sync	**88.11**	**93.36**	**95**	**94.51**
5-gram add-1	13.99	42.74	51	37.27
5-gram gt	13.99	42.74	51	37.27

Table 4.4: The table illustrates the performance achieved by the preliminary experiments on English. We show experiments on the four sets of confusibles for the following 3-gram, 4-gram, and 5-gram SLMs: (1) continuous back-off (cont), (2) synchronous back-off (sync), (3) add-1 smoothing (add-1), and (4) Good-Turing smoothing (gt). Results are shown by their accuracy (%).

by PS1, PS2, and RQ1 to RQ4. From the results (both Dutch and English) we have drawn four conclusions. They are listed below.

First, for making a decision between alternative sequences, smoothing methods do not help to arrive at a better performance. Smoothing methods do not harm the performance either. We stress that, when using smoothing methods, the relative order of the probabilities between the sequences remains the same. Thus, smoothing does not influence the decision made.

Second, as we see in Table 4.3 the weighted linear approach outperforms the uniform linear approach on all confusible sets. We may conclude that the contribution from the n-grams with large n is overruled by the probabilities of the n-grams with smaller n in the uniform linear method. This causes a bias towards the more frequent words, compounded by the fact that bigrams, and unigrams even more so, are less sparse and therefore contribute more to the total probability of the entire sequence than the trigrams do.

Third, for continuous back-off versus synchronous back-off, it is detrimental to compare different-size n-grams on the same position in different alternative sequences. In the case of continuous back-off, if on one of the alternatives a lower-order model is used it will generally receive a larger score. Avoiding to compare scores from different distributions is why *synchronous back-off* works well.

Fourth, while larger n-grams have detrimental effects on the other methods, they do not harm the robust *synchronous back-off* method much. They can help if the typical context for the alternatives is different. If possible, we prefer larger n-grams, as they provide more precise statistics.

4.2.3 Impact on our Work

As stated above, the preliminary experiments guided our further investigation. First, they resulted in a *modular* approach (described in Chapter 3).

Second, the experiments showed the power of synchronous back-off. Synchronous back-off avoids comparing scores from different distributions. So, from here on, we will exclusively employ the synchronous back-off method.

4.3 Language-Model Environment

The language-model environment used in this chapter can be characterised by two aspects: (1) the usage of unbounded size n-grams, and (2) not using annotations. We call the unbounded size n-grams ∞-grams. All experiments are

performed with synchronous back-off. For a proper reading we provide two practical details that are essential for our implementation.

The first detail is that the actual implementation of synchronous back-off is a reversed one compared to its description above. We do not start at the largest possible n-gram and reduce its size until a probability can be assigned. Instead, we start with the smallest size n-gram (i.e., the unigram) and step by step increase the n-gram size until we can no longer assign a non-zero probability. At that point, we use the n-grams one size smaller.

The second detail of the implementation is that we only calculate the probabilities of n-grams that have some overlap with the focus position. As the focus position contains the only difference between the generated alternatives, n-grams that do not cover the focus position will be equal among alternatives. Thus, the relative differences in score between the alternatives will stay the same.

$$P_n(w_1, \ldots, w_m) = \prod_{i=1}^{m} P_n(w_i | w_{i-(n-1)}, \ldots, w_{i-1}) \tag{4.7}$$

The probability of a sequence is assigned using Equation 4.7[12]. The calculation of each positional probability in the sequence is done separately (see Equation 4.8). With this equation we calculate the probability of an n-gram at position x, i.e., the probability of part of the sequence. We perform these calculations by looking up the counts of the relevant n-gram and corresponding $n-1$-gram in the suffix array. Using these counts we calculate $P_n(w|w_{-(n-1)}, \ldots, w_{-1})$. By combining the positional probabilities, we determine the sequential probability $P_n(w_1 \ldots w_m)$.

$$P_n(w|w_{-1}, \ldots, w_{-(n-1)}) = \frac{\#(w_{-(n-1)}, \ldots, w)}{\#(w_{-(n-1)}, \ldots, w_{-1})} \tag{4.8}$$

We remark that for all probability calculations we use the logarithmic values of those probabilities as much as possible (see Equation 4.9). This prevents underflow of the floating point numbers used in calculating fractions and increases the precision of the calculations. It is a well known technique and it is used in most language model implementations, including SRILM (Stolcke 2002).

$$P(x) \times P(y) \equiv \log^{-1}(\log P(x) + \log P(y)) \tag{4.9}$$

[12] This formula was already given above in Equation 4.2 on Page 55.

For Equation 4.8 we need to look up the counts of n-grams. In order to be able to produce these counts for any size n we use a suffix array to look up the counts. The suffix array is built on the training part of the corpus after conversion to a string where each type is mapped to a unique integer. This conversion step speeds up the operations in the suffix array considerably, as the basic operations on integers are faster[13]than the basic operations on strings. We note that integers also use less storage than strings.

We summarize the operations performed by this version of the language-model environment as follows. First, a suffix array data structure is built on the training part of the corpus. The training consists of plain sequences (such as natural language sentences), i.e., material without any added annotations. Second, the language model is applied to the set alternatives, using the formulas in Equation 4.8 and Equation 4.7. The application of the language model is done using synchronous back-off, in order to use the largest n-gram size applicable to the decision throughout the experiments. Finally, the output is a set of score per set of alternative sequences. Each alternative sequence has its corresponding score. The post-processing and evaluation is done afterwards as described in Chapter 3.

4.4 Experiments

In this section we describe the results of the experiments as described in Chapter 3. Starting with the testing material, we generate alternative sequence sets. For each of the alternative sequence sets we assign a score to each member. This score assignment is performed with synchronous back-off with an unbounded size n-gram. Subsequently, we choose the alternative with the highest score.

If the choice coincides with the gold standard text we count the choice as correct. If it does not coincide then the system generated an incorrect choice. That is, we assume the training and testing material to be free of errors. We may take Reynaert's (2005) estimate that 0.25% of all tokens in a corpus of the type we use represent a non-word error. We take this as an indication that similar error levels can be expected to occur for our three alternative selection tasks.

---

[13] In particular, this statements holds true for the basic operation *compare*. Comparison operations on integers are faster than comparison operations on strings. For a string these operations use $O(n)$ time, where n is the length of the string. For integers, these operations are done in $O(1)$ time. Since almost all operations on the suffix array use mainly comparisons between elements of the array we gain speed. In particular operations concerning construction and searching for a pattern use these comparisons.

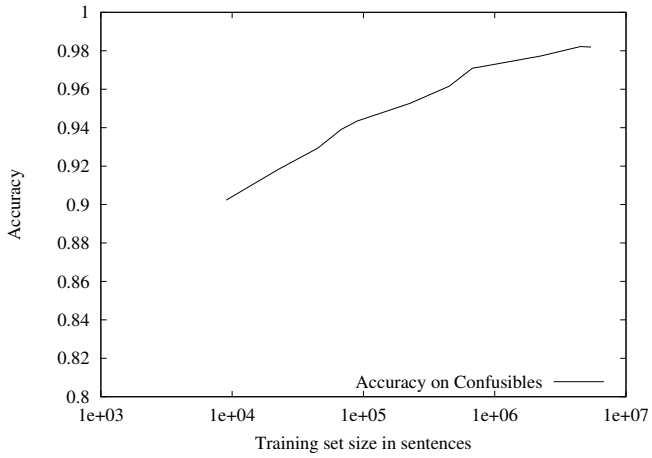


Figure 4.4: Learning curve for the confusible problem. Results obtained with synchronous back-off and without annotations. Y-axis denotes the accuracy and the X-axis the number of sentences used for training. The X-axis is logarithmic.

In our experiment we record the errors. Ideally, we may expect zero errors, because our point of departure is the gold standard text. Since we deal with insufficient data, the language model will make errors.

We relate the number of errors to the training size. As errors can be caused by insufficient data, so we also investigate the effect of the amount of training data. As a next step we interpret and analyze the results. We relate these to the n-gram size used for making the classification.

We investigate our three alternative selection problems, viz. confusibles, verb and noun agreement, and prenominal adjective ordering. For all three problems we report on (1) the influence of the amount of training material on the performances, (2) the average size n-gram used, and (3) the relation between the n-gram size used for a decision and the performance.

4.4.1 Results on Confusibles

The execution of our series of experiments on the confusibles was as follows. In the above, we stated that we are interested in the relation between the size of the training material and the performance. Therefore we present the results in the form of learning curves. The learning curves start at $9,000$ sentences of training material. The end-point of the learning is reached when 90% of the

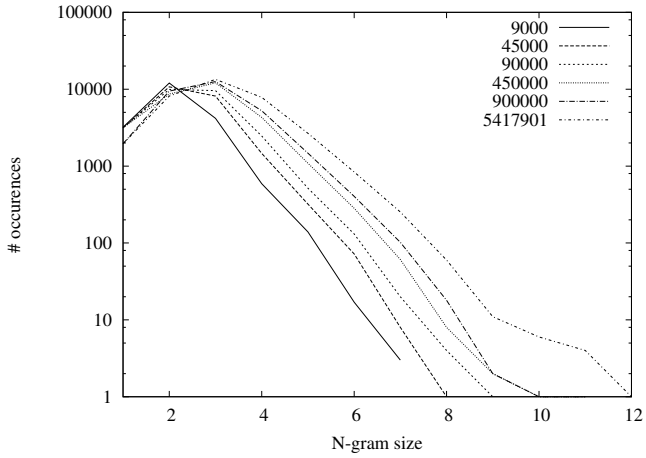


Figure 4.5: Occurence graph for the confusibles problem showing, for each training set size, the number of times each n-gram size was used in making the alternative selection. Results obtained with synchronous back-off and without annotations. The Y-axis is logarithmic.

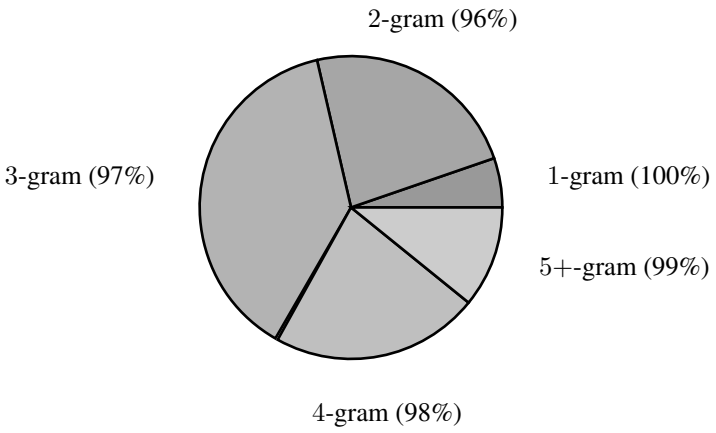


Figure 4.6: A pie chart showing how often which n-gram size was used for the classification on the confusibles problem and the corresponding accuracy. The pie chart is drawn for the full size training set.

corpus is used for training, as 10% is reserved for testing. This learning curve is shown in Figure 4.4.

In Figure 4.4 we see that the performance achieved roughly log-linear compared to the size of the training set and near the full amount of training material it starts to flatten. With our smallest training set (i.e., consisting of 9,000 sentences) we achieved an accuracy of 90.23%. With the full training set (i.e., the entire BNC minus the testing part) we achieved an accuracy of 98.19%. Our performance can be compared to the performance achieved in the literature. Golding and Roth (1999) achieved as a best result a score of 96.6% on the same confusible sets. Here we mention that other researchers with other confusible sets achieve a higher score than 96.6%, cf. Van den Bosch (2006a), Stehouwer and Van den Bosch (2009). Yet, we are not aware of any publication with a score higher than 98% on these confusible sets. We remark that, as the system is still increasing in performance at the end of the learning curve it is unlikely that this is the upper limit of performance.

The size of the n-gram used for classification is also measured in our experiments. Below we will report on (1) the average size of the n-grams and (2) on the distribution of the n-gram sizes. The average size of the n-gram used for classification increases with the amount of training material. At the smallest training set (i.e., 9,000 sentences) the average n-gram size found by the system is 2.1. At the largest training set the average n-gram size found by the system is 3.1. The distribution of the n-gram sizes can be seen in Figure 4.6 where we show a pie-chart of the number of occurrences for the n-gram sizes compared to the performance. We observe that bigrams, trigrams and 4-grams are used in 84% of the cases. We note that the performance unigrams is as high as those specific instances coincide with the majority class.

4.4.2 Results on Verb and Noun Agreement

The execution of our series of experiments for the verb and noun agreement is performed analogously to the experiments in Subsection 4.4.1. The learning curve for our results on verb and noun agreement is shown in Figure 4.7.

The performance achieved roughly log-linear compared to the size of the training set and near the full amount of training material it starts to flatten. With our smallest training set (i.e., consisting of 9,000 sentences) we achieved an accuracy of 55.87%. With the full training set (i.e., the entire BNC minus the testing part) we achieved an accuracy of 80.73%. Our performance can be compared to the performance achieved in the literature. On the one hand, Chodorow and Leacock (2000) reported a classification precision of 77.9%. On the other hand, Schaback and Li (2007) achieved a suggestion accuracy of 90% when us-

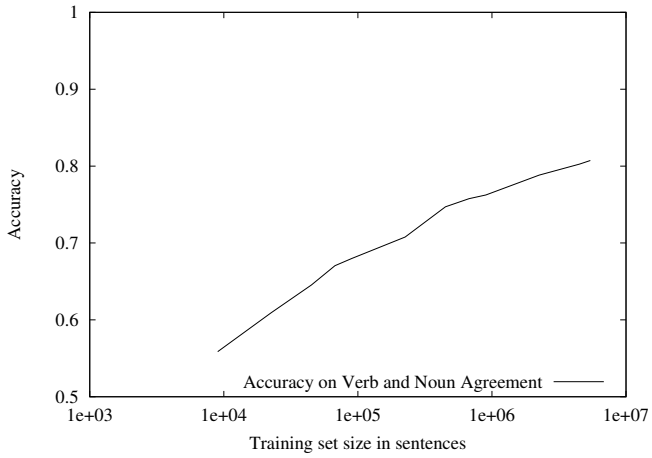


Figure 4.7: Learning curve for the verb and noun agreement problem. Results obtained with synchronous back-off and without annotations. Y-axis denotes the accuracy and the X-axis the number of sentences used for training. The X-axis is logarithmic.

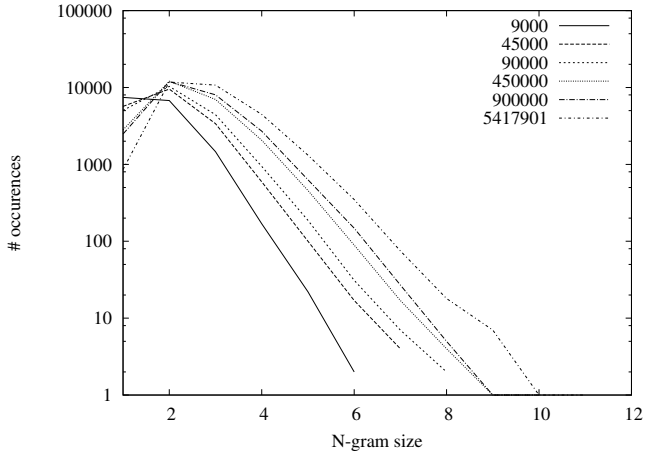


Figure 4.8: Occurence graph for the verb and noun agreement problem showing, for each training set size, the number of times each n-gram size was used in making the alternative selection. Results obtained with synchronous back-off and without annotations. The Y-axis is logarithmic.

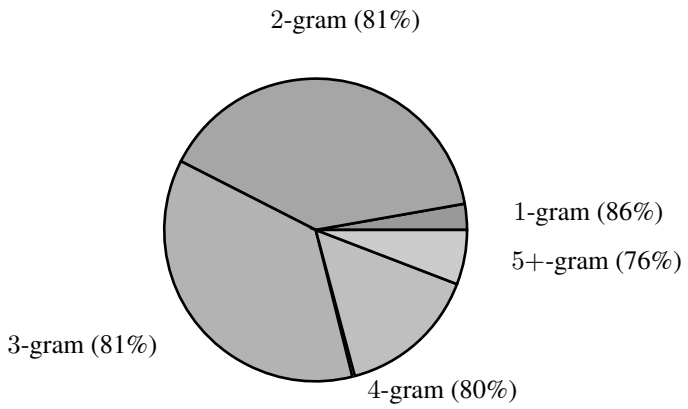


Figure 4.9: A pie chart showing how often which n-gram size was used for the classification on the verb and noun agreement problem and the corresponding accuracy. The pie chart is drawn for the full size training set.

ing their 1-best system[14]. So, our performance falls within the range of reported performance in the literature even though our system consists of a generic alternative sequence selector not specialized on this task.

The size of the n-gram used for classification is also measured in our experiments. Below we report again on (1) the average size of the n-grams and (2) the distribution of the n-gram sizes. The average size of the n-gram used for classification increases with the amount of training material. At the smallest training set (i.e., 9,000 sentences) the average n-gram size found by the system is 1.6. At the largest training set the average n-gram size found by the system is 2.8. The distribution of the n-gram sizes can be seen in Figure 4.8 where we show a plot of the occurence graph of the used n-gram sizes for each training size. We remark that the n-gram size used increases with the training size, as one would expect. These observation are similar to the ones on the confusible task in Subsection 4.4.1. In Figure 4.9 we relate the performance to the size of the n-gram used for classification. We observe that bigrams, trigrams and 4-grams are used in 91% of the cases.

[14] They also reported results on 2-best, 3-best, and so on systems. These systems achieved a lower (2-best) and lower (3-best) accuracy, as the extra predictions are counted against the accuracy. In effect, the maximum accuracy of the 2-best system is 50%. Their accuracy was measured by comparing the output of the system to a manual correction of the error (from a list of errors, i.e., no error detection was performed).

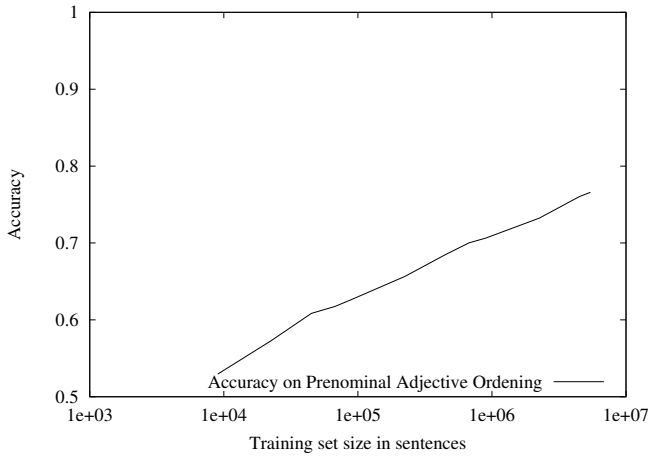


Figure 4.10: Learning curve for the prenominal adjective order problem. Results obtained with synchronous back-off and without annotations. Y-axis denotes the accuracy and the X-axis the number of sentences used for training. The X-axis is logarithmic.

4.4.3 Results on Prenominal Adjective Ordering

The execution of our series of experiments for the prenominal adjective ordering is performed analogously to the experiments on confusibles in Subsection 4.4.1 and the experiments on verb and noun agreement in Subsection 4.4.2. The learning curve for our results on prenominal adjective ordering is shown in Figure 4.10.

The performance achieved roughly increases linearly compared to the log of the size of the training set. With our smallest training set (i.e., consisting of 9,000 sentences) we achieved an accuracy of 52.94%. With the full training set (i.e., the entire BNC minus the testing part) we achieved an accuracy of 76.59%. Our performance can be compared to the performance achieved in the literature. Malouf (2000) achieved as a best result a score of 92% on the same task. Malouf (2000) also achieved a prediction accuracy of 75.57% using a straightforward word trigram model. Shaw and Hatzivassiloglou (1999) reported an achieved accuracy of 86.17% on newspaper data using a multi-cluster model and of 75.41% using a direct evidence model. So, our results are on the low end of the reported results in the literature. However, we performed similarly to the other non-specialised models which Malouf (2000) and Shaw and Hatzivassiloglou (1999) reported on, i.e., models similar to n-gram mod-

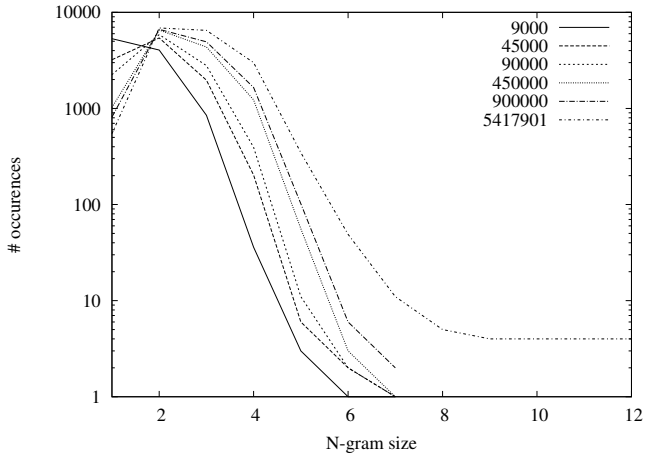


Figure 4.11: Occurence graph for the prenominal adjective order problem showing, for each training set size, the number of times each n-gram size was used in making the alternative selection. Results obtained with synchronous back-off and without annotations. The Y-axis is logarithmic.

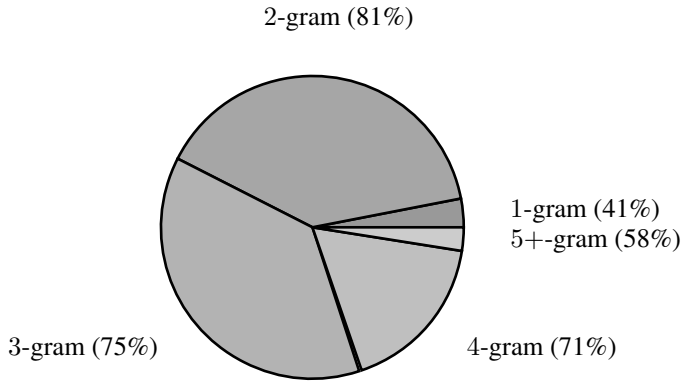


Figure 4.12: A pie chart showing how often which n-gram size was used for the classification of the prenominal adjective ordering problem and the corresponding accuracy. The pie chart is drawn for the full size training set.

els. Our tentative conclusion here is that prenominal adjective ordering is a task which benefits greatly from specialised models[15].

[15] We have discussed these specialised models in Section 2.3.

The size of the n-gram used for classification is also measured in our experiments. Below we report on (1) the average size of the n-grams and (2) the distribution of the n-gram sizes. The average size of the n-gram used for classification increases with the amount of training material. At the smallest training set (i.e., 9,000 sentences) the average n-gram size found by the system is 1.57. At the largest training set the average n-gram size found by the system is 2.8. The distribution of the n-gram sizes can be seen in Figure 4.11 where we show a plot of the occurence graph of the used n-gram sizes for each training size. We remark that the n-gram size used increases with the training size, as one would expect. In Figure 4.12 we relate the performance to the size of the n-gram used for classification. We observe that bigrams, trigrams and 4-grams are used in 94% of the cases. In case of prenominal adjective ordering unigrams perform particularly bad, due to the fact that with unigrams the adjecency adjectives is not taken into account.

4.5 Answers to RQ1 and RQ2

In this chapter we investigated a language-model environment without annotation that uses n-grams of unlimited size. The investigations provide a basis to answer our research questions RQ1 and RQ2. Below we repeat RQ1 and RQ2. They are followed by our answers.

> **Research Question 1:** Is there a need to predetermine or limit the size of the n-grams used in language models? Is there an inherent advantage or disadvantage to using a fixed-size n?

In the general case we are facing an ambiguous answer: restricting the size of the n-gram is a disadvantage, however it does not impair performance. In the three alternative sequence selection tasks we observe that the number of times n-grams larger than 5 are used are low. So, even though the larger size n-gram helps, it does not occur often enough to impact performance positively. However, restricting the size of the n-grams can be done in two ways: 1) the size of the n-grams can be fixed ahead of time, or 2) the size of the n-grams can be restricted based on the specific situation they occur in, as is done with synchronous back-off.

From our preliminary results described and discussed in Section 4.2 we learn that distributions of different sized n-grams produce incomparable scores, and distributions of the same sized n-grams produce comparable scores. Using synchronous back-off we ensure that we always generate comparable scores.

Owing to synchronous back-off we were able to show that there is no need to predetermine and restrict the size of the n-gram in advance. However, the

sparseness of the training data used will automatically limit the size of the n-gram that synchronous back-off uses for decisions. The optimal size of the n-gram used for a decision between alternative sequences can differ for each position of each set of alternative sequences. From these observations we may conclude that there is an advantage when the size of the n-gram us not predetermined.

> **Research Question 2:** If the size of the n-grams is not fixed in advance, how can we still generate comparable distributions when we select among alternative sequences?

The synchronous back-off method ensures that the score of all alternatives in a set of alternative sequences is calculated using comparable distributions. We can generate comparable distributions by synchronizing the sizes of the n-grams used for the calculations. This process is described in detail in Section 4.2.

4.6 Chapter Conclusion

In this chapter we investigated a language-model environment that can be characterised as a word-based language model without any additional linguistic annotation. We focussed on increasing flexibility, and on automatically determining an optimal n-gram size to use on a position-by-position basis.

We discussed the results from the preliminary experiments. Synchronous back-off performs a back-off step to the same order n-grams at each position within a set of alternative sequences. Synchronous back-off avoids comparing scores from different distributions. Using synchronous back-off leads to better alternative sequence selection.

The experiments performed on the three alternative selection problems led to detailed results for the language-model environment described in this chapter. The results are compared with respect to the n-gram size used. Here we observe that decisions made on larger size n-grams are better. However, such large n-grams do not occur sufficiently to increase the overall performance. More than 80% of the classifications are performed with bigrams, trigrams, or 4-grams (as also visualized in the pie charts in Figures 4.6, 4.9, and 4.12). Specifically in case of the confusibles problem 84% of decisions are made by bigrams, trigrams, or 4-grams. For the verb and noun agreement problem this is the case for 91% of decisions, and for the prenominal adjective ordering this is the case for 94% of decisions. Yet, from the above results we may conclude that in some cases a model using flexible sized n-grams for decisions comes closer to the ideal model.

Chapter 5

Models with Local Annotation

This chapter describes an LME that is to be characterised as a model with local annotation. It means that we focus on the effect that adding such information has on the language-model environment. Local annotation is annotation that is added to each word in the sentence, which depends only on the word itself and its local context. The LME described in this chapter also includes the aspects discussed in Chapter 4, i.e., the ∞-gram and synchronous back-off. Using this enhanced LME and its corresponding LM we perform our experiments. There are many possible types of local annotation such as named entities, part-of-speech, and morphological annotation. In this chapter we focus on part-of-speech annotation. We examine RQ3 and RQ4 with respect to the local annotation model for the three alternative sequence selection problems. Below we re-iterate both RQs and remark that for this chapter 'annotations' refer to 'local annotations'.

> **Research Question 3:** Is there a benefit to including annotations in the language model, measured as a better performance on alternative sequence selection tasks?

> **Research Question 4:** Is there a difference in performance on alternative sequence selection tasks when using human-designed annotations compared to machine-generated annotations?

The course of this chapter reads as follows. We start by describing four issues related to part-of-speech annotations. This is done in Section 5.1.

article	noun	verb	prep.	noun	punct.
The	robots	walk	in	space	.

Figure 5.1: An example with fabricated part-of-speech tags.

In Section 5.2 we complement the language-model environment as described in Chapter 3 with (1) a precise description of the use of human-designed part-of-speech tags, and (2) the generation and use of machine-derived ones. This section uses and expands the environment as described in Section 4.3. So, all information given there still applies.

In Section 5.3 we describe our experiments on the three alternative sequence selection problems. Next to the three problems, in this chapter we investigate the influence of adding part-of-speech tags. Moreover, we contrast the performance of human-designed part-of-speech tags to the performance of machine-derived tags.

The results of the experiments are analysed with respect to RQ3 and RQ4. In Section 5.4 we use this analysis to formulate our answers to RQ3 and RQ4s for the models with local annotation. In Section 5.5 we provide a summary and give our conclusions.

5.1 Part-of-Speech Annotation

A frequently used type of local annotation is part-of-speech annotation. For each word within a sequence the tag denotes its morphosyntactic category. The categorisation is based on the syntactic and sometimes morphological behaviour of the word in its context. Part-of-speech tags are generally defined by linguists and dependent on locally word context. Typically, these annotations are assigned by hand to a reference corpus. Each word in a sequence has such a part-of-speech tag in a tagged corpus. For a fabricated example we refer to Figure 5.1. The sequence *The robots walk in space .* is mapped to a labeling consisting of an article, two nouns, a verb, a preposition, and a punctuation tag.

Part-of-speech tags form a useful annotation layer supporting the linguistic analysis of a text or a corpus. Their usefulness is in part due to the disambiguation of the function of a wordform, e.g., *walk* can be both a noun and a verb. Sometimes, part-of-speech tags are used in the automatic analysis of a text (e.g., by participating as a part of a pattern, cf. Sun et al. 2007). A part-

of-speech tag can also be used as a building block to construct other, more complex annotation layers. An example of a complex annotation layer is a syntactic parse (see Charniak et al. 1996, Hall et al. 2007, Nivre et al. 2007b, Nivre and McDonald 2008). A parse is an analysis of the syntactic structure of a sequence.

Below, we briefly discuss (1) part-of-speech tags as defined by human experts, (2) part-of-speech tags as derived by a computer, (3) the evaluation of the quality of computer-derived tags, and (4) the automatic assignment of part-of-speech tags to new sequences.

5.1.1 Human-Defined Part-of-Speech Annotation

Over the years, many diverse sets of part-of-speech tags have been defined for many different languages. Subsequently, the elements of these tagsets were assigned manually to the words in a corpus. From the many tagsets constructed by hand we single out three of them for discussion in this subsection: (1) the Penn Treebank tagset (see Marcus et al. 1993), (2) the CGN tagset as used in the Corpus Gesproken Nederlands (see Van Eynde et al. 2000, Van Eynde 2004), and (3) the CLAWS5 tagset as used in the British National Corpus (see Leech et al. 1994).

The Penn Treebank (PTB) is a corpus consisting of full parse trees, including part-of-speech tags, of mainly newspaper text. The PTB also contains general English texts (Brown) and dialog texts (ATIS). Marcus et al. (1993) describe the process of creating the tagset. It is a significantly reduced version of the Brown corpus tagset (see Kučera and Francis 1967, Francis and Kučera 1982). It consists of 36 part-of-speech tags and 12 other tags (i.e., for punctuation and currency symbols). The tagset uses monolithic tags, such as *NN* for a proper noun. Monolithic tags are tags that are non-divisible and that consist of one element.

The CGN tagset for the Dutch language is developed by Van Eynde et al. (2000). It is quite different from the Penn Treebank tagset by its modularity and detailed tags. The tagset consists of more than three hundred different applicable[1] tags, which are described in detail in Van Eynde et al. (2000) and Van Eynde (2004). The modularity of the tagset is an important element. Modularity means that a tag consists of multiple independent elements which can be easily viewed and queried separately. An example is *N(soort,ev,basis,zijd,stan)*[2]

[1] Here, applicable refers to the way words are used in the Dutch language, e.g., diminutive and non-neuter will not occur as a combination.

	...	-2			-1			1	...
			,	the	.	of	to		
,			0	0	0	8	20		
the			4,244	11	0	6,159	1,959		
.	...	...	0	0	1	8	19	...	...
of			218	0	0	1	0		
to			570	0	0	2	2		

Figure 5.2: An partial example of co-occurrence vectors. We show position -1 for the co-occurrences of the top five most frequent words in the WSJ part of the PTB.

for a singular, basic proper noun that is non-neuter. One of the elements in this example tag is *ev* (i.e., enkelvoud, which means singular). The hierarchical element at this position could also be *mv* (i.e., meervoud, which means plural). Such a tagging system enables the linguist to compile statistics and search for patterns for specific properties.

A slightly larger tagset than the Penn Treebank tagset is the CLAWS5 tagset (Leech et al. 1994). This tagset is used in the British National Corpus. The CLAWS5 tagset consists of 57 monolithic tags. Two examples are as follows. First, one of the tags is *AJS* for a superlative adjective, e.g., best, oldest, largest, etc. Second, one of the tags is *PNP* for a personal pronoun, e.g., you, them, etc. Leech et al. (1994) describe the basic process used for tagging the British National Corpus with these tags. This process was automatic and supervised by linguists on a sampling of the output, an error rate of 1.5% was found on the tagger output. We mention this tagset here as it is the tagset we employed in our experiments (cf. Chapter 3). Our choice of this tagset follows from the choice of our corpus, as this is the human-designed tagset for the BNC.

5.1.2 Machine-Derived Part-of-Speech Annotation

In the literature several approaches have been proposed to derive tags on sequences of words automatically. The derived tagsets are assumed to resemble part-of-speech tags. Here we will discuss two approaches: (1) the clustering of

[2] The annotation is as follows. *N* stands for a proper noun, *soort* stands for *soortnaam* (type name), i.e., a name that is not capitalised, e.g., zondag (sunday) or maandag (monday), *ev* stands for *enkelvoud* (singular), *basis* stands for the basic form of the noun, i.e., not its diminutive form, *zijd* stands for *zijdig* (gendered), i.e., the noun has a gender (in Dutch, nouns can be gendered or neuter; however, nouns referring to an object or a concept do not have a specific gender as far as grammatical use is concerned), *stan* stands for *standaard* (standard), i.e., the noun uses the standard conjugation. Van Eynde (2004) describes in detail all the possible elements of the tagset (the article is in Dutch).

co-occurrence vectors (Schütze 1993), and (2) graph colouring of neighbourhood graphs Biemann (2007).

The work by Schütze (1993) is a seminal work on the unsupervised part-of-speech tagging. This means that the tagset and the tagging are both decided upon by an algorithm. Schütze (1993) represents each frequent word as a co-occurrence vector of its neighbouring frequent words. A word is defined as frequent if its number of occurrences has passed a previously set threshold. A co-occurrence vector consists of elements that are vectors themselves. Each of these elements represents the number of occurrences of each frequent word at a specific offset position. For instance, when the offset position is -3, i.e., the content of the offset position -3 is compared to the frequent word that the vector represents. To substantiate the idea, consider the sentence *The robots walk in space*. If the focus position is on *in*, then for the offset vector -3 the value corresponding to the combination *The...in* is increased by 1. We illustrate the concept of co-occurrence vectors in Figure 5.2. Clusters of these co-occurrence vectors then represent a single part-of-speech tag[3]. The number of part-of-speech tags depends on the number of clusters found.

Biemann (2007) introduced a more recent unsupervised learning approach to part-of-speech tagging, called CHINESE WHISPERS (the name is derived from a children's game). CHINESE WHISPERS is a non-deterministic graph-colouring approach to clustering. In this approach, words are represented as nodes, and the edges between the nodes are determined by a threshold on the number of shared (co-occurrence) neighbours. Once a graph is constructed, all nodes are assigned a random colour. After the construction phase the algorithm iterates many times over all nodes in the graph. Each round this is done in a different, random order. When a node is visited it is assigned the majority colour at that position based on the incoming edges and the colour of the nodes those edges connect to. This graph-colouring algorithm is self-terminating[4] and automatically determines the number of clusters used. When it terminates it has assigned the same colour, or part-of-speech tag, to strongly-connected subgraphs of the neighbourhood graph. A strongly-connected subgraph is a part of a graph of which its internal connections result in direct connections between all vertices of the subgraph.

Machine-derived tagsets are characterised by the fact that they arise from the data itself. The advantage is that no expensive human annotator or linguistic researcher is needed to generate a tagging. However, it is not clear whether such an unsupervised method is able to provide a tagging that is similar to one obtained using a supervised method on a human-designed tagset. The evalua-

[3] The clustering itself can be done with any vector-based clustering algorithm, such as k-means.

[4] As soon as the difference in the labeling of the nodes between rounds is $< \epsilon$ the algorithm terminates.

tions present in the literature do not fully address this problem. We will go into more detail on this in Subsection 5.1.3.

5.1.3 Evaluation of Machine-Derived Annotations

We start by listing two assumptions:

1. Assume that we are in possession of a set of machine-derived annotations on a corpus.
2. Assume that the corpus is also annotated by a human-designed tagset (e.g., part-of-speech tags).

Now the question arises, how should we evaluate the set of machine-derived annotations?

In the literature, a common approach is qualitative analysis[5] (cf. Schütze 1993), which is mostly performed manually. However, manually looking whether a set of words for a specific tag in a context looks right becomes quickly unfeasible with any decent-sized corpus and it is highly subjective. So, we should attempt to perform the evaluation automatically and objectively.

Circumventing the problems of using qualitative methods we take a quantitative approach to the evaluation of machine-derived part-of-speech tags. We compare the machine-derived part-of-speech tags with the gold standard human-designed annotation. There are two common metrics to perform such comparisons: (1) directly comparing the machine-derived tags to the human-designed tags in terms of shared distribution, i.e., using concepts such as precision, recall, and F-score, and (2) measuring the predictability of the human-designed tags by the machine-derived tags in terms of the reduction in perplexity.

For the first evaluation we employ the F-score measure introduced by Van Rijsbergen (1979). This measure is calculated by combining the the scores for precision and for recall. In the equations for precision and recall the terms TP, FP, TN, and FN stand for true positive, false positive, true negative, and false negative, respectively. We illustrate the classes of true and false positives and negatives in Figure 5.3. The positive class is represented as the target class in the explanations below. This approach generalises to a multi-class problem, where for calculating the F-score for a class that class is the positive target and all the other classes combined are the negative target.

[5] In qualitative analysis a deep, manual inspection of the material is made in order to come to a conclusion. Qualitative analysis stands in contrast to quantitative analysis, in which case the analysis is done based on measurable aspects.

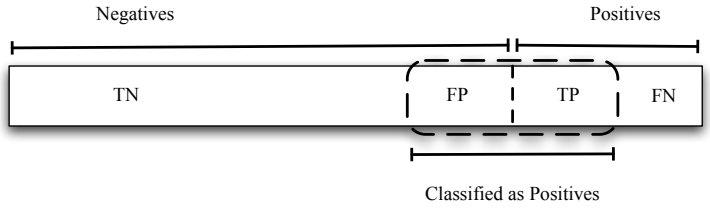


Figure 5.3: Illustration of the areas that are true negatives (TN), false positives (FP), true positives (TP), and false negatives (FN). Illustration adapted from Figure 1 in Reynaert (2008).

true positive An element classified correctly in the target class is a true positive.

false positive An element classified erroneously as belonging to the target class is a false positive.

true negative An element classified correctly as not in the target class is a true negative.

false negative An element classified erroneously as not belonging to the target class is a false negative.

Precision (P) defines the number of correctly identified positive examples compared to the total number of positive examples identified. We show precision in Equation 5.1.

$$P = \frac{TP}{TP + FP} \tag{5.1}$$

Recall (R) defines the number of correct positive examples identified compared to the total number of correct positive examples. We show recall in Equation 5.2.

$$R = \frac{TP}{TP + FN} \tag{5.2}$$

We define the F-score as shown in Equation 5.3, this formula is derived from the formula introduced by Van Rijsbergen (1979).

$$F = \frac{2 \times P \times R}{P + R} \tag{5.3}$$

Traditionally the measure F defines a weighting between precision and recall using the parameter β. For β we will use the value 1 resulting in the harmonic mean between precision and recall. In the equation above we show the simplification resulting from removing the (effectively unused) factor β from the formula.

The second approach was introduced by Freitag (2004) and is called *cluster–conditional tag perplexity*[6]. The approach was also used by Biemann (2007) to evaluate the output of his graph-based part-of-speech tagger. The measure is based on a combination of the *entropy* (or *Shannon information*) I_X and the *mutual information* of the distribution of clusters and human-designed tags M_{XY}.

Mutual information is a measure of mutual dependence of two variables. We define the mutual information M_{XY} of two variables X and Y as in Equation 5.4. In Equation 5.4 $P(x)$ is the chance of x and $P(x, y)$ is the chance of x and y. We note that the MI metric becomes negative if the separate probabilities for x and y are much larger than the conditional probability $P(x, y)$.

$$M_{XY} = \sum_{XY} P(X, Y) \log_2 \frac{P(X, Y)}{P(X)P(Y)} \tag{5.4}$$

The mutual information metric is used in a variety of NLP investigations, from compound extraction (Wu and Su 1993), via word association (Church and Hanks 1990), and via parsing (Magerman and Marcus 1990), to evaluation of unsupervised tagging (Freitag 2004).

Entropy, or Shannon information, is a measure of the inverse likelihood of a sequence. We define the entropy I_X over the variable X as in Equation 5.5.

$$I_X = -\sum_{x} P(x) \log_2 P(x) \tag{5.5}$$

Freitag (2004) combined Equations 5.4 and 5.5, calculating the reduction of the perplexity (R) over the set of human-designed tags (T) with respect to the machine-derived set of annotations (C).

$$R_{TC} = I_T - M_{TC} \tag{5.6}$$

[6] Perplexity usually refers to a measure of surprisedness when observing the data compared to some model. In this case the surprisedness of observing one set of tags compared to another set of tags on the same sequences.

In effect the R_{TC} determines the perplexity for the classification of the human-designed tagset T (i.e., I_T) resulting from the presence of the machine-derived annotation C measured by M_{TC}. The perplexity for the human-designed tags given an machine-derived annotation is a better evaluation of machine-derived annotations than the F-score (Freitag 2004, Biemann 2007) as it more accurately measures the prediction value of the annotation.

5.1.4 Applying Part-of-Speech Tags Automatically

Human-defined tagsets are not typically applied to new sequences by hand as this is unfeasible for very large amounts of text. Typically a supervised automatic tagging system is trained on a small amount of manually tagged text and then applied to new text automatically. Over the years several approaches for supervised automatic tagging have been developed, we mention three of them: (1) using a statistical model, (2) using rule induction, and (3) using a memory-based model.

The most popular approach is using a statistical model for the assignment of tags. Like all three approaches, the statistical model is trained on some set of correctly-tagged training material. This approach is used in many different taggers, such as the Xerox tagger (Cutting et al. 1992), TnT (Brants 2000), and many others (Steetskamp 1995; Ratnaparkhi 1996; Màrquez and Padró 1997). During training the model stores conditional probabilities of n-gram sequences of tags and input text. After training, the choice for a specific tag is made on the basis of all possible sequences of tags given the input sentence. Out of these possible tags the sequence that fits the sentence, given the model, best is chosen. For selection of the most likely path through the lattice of possibilities, the Viterbi (1967) algorithm is generally employed. The algorithm is a dynamic programming approach to finding the optimal path through a lattice of possible paths where, in this case, a path describes a part-of-speech sequence.

A second approach to supervised automatic tagging is by the use of rule induction. In the field of rule induction the learning system tries to derive a set of rules for some task. The method developed by Brill (1992) is prototypical for this approach on part-of-speech tagging. Brill (1992) introduced the transformation-based learning part-of-speech tagger. In the training phase of this approach the tagger starts with a base state of the corpus. In this base state each word is tagged with its most frequent tag based on the training data. Thereafter, transformation rules are learned and applied to the tags to improve the tagging as compared to the original training data. The most promising rule is learned first, i.e., the rule that provides for the greatest improvement. These rules are context dependent. An example is: *change tag a to tag b when: one of the three preceding words is tagged z.*

(Brill 1992). When applying the learned ruleset to a new sentence that sentence is first tagged with the most frequent tags for each word. After that all the learned transformation rules are applied, in order, to the sentence in order to produce the tagging.

The third approach is that of using a memory-based model. An example of this approach is the MEMORY-BASED TAGGER (MBT) as described by Daelemans et al. (1996a,b), Zavrel and Daelemans (1999), and Van den Bosch et al. (2007). The MEMORY-BASED TAGGER uses the TIMBL memory-based learner (Daelemans et al. 2010) to perform the classifications. TiMBL is based on the k-nearest neighbour approach (Cover and Hart 1967, Dudani 1976). In the k-nearest neighbour approach all instances are loaded into memory in the training phase; in the testing phase the distance from the new instance to all learned instances is calculated. The k closest instances are then used for classification by using weighted voting. MBT learns memory-based taggers trained on user-selected features. It automatically creates two taggers, one for known words and one for unknown words. The tagger for the unknown words is based on data of infrequent words. Zavrel and Daelemans (1999) argue that these infrequent words are more likely to resemble unseen words than the very frequent words of the language.

An important advantage of memory-based learning on language-based tasks as opposed to, for instance, a Maximum Entropy model (Guiasu and Shenitzer 1985) is that even though memory-based learning generalizes, the model does not forget local exceptions[7] (Nøklestad 2009, Section 2.5). As Daelemans et al. (1999) argue, forgetting exceptions is harmful in natural language processing models as language contains a large number of exceptions to the rules.

5.2 Language-Model Environment

The LME used in this chapter and its corresponding LM can be characterised by two aspects. First, the LM is a model with unbounded n-grams (i.e., the ∞-gram) controlled with synchronous back-off. Second, the model employs an annotation layer. The annotation layer consists of part-of-speech information. The annotation layer is stored inline in the training data, with the annotation directly after the word it applies to. We will perform two sets of experiments, namely (1) with human-designed part-of-speech tags, and (2) with machine-derived part-of-speech tags.

[7] The sensitivity to local exceptions and the amount of generalisation are opposed. The balance between the two can be influenced with the k value. A larger value of k results in more generalisation and less sensitivity to local exceptions.

Below we describe our experimental setup. As in Chapter 4 we only cover what differs from the description in Chapter 3. First, we discuss the human-designed part-of-speech tags and the development of the machine-derived part-of-speech tags in Subsection 5.2.1. Second, we discuss the evaluation of the machine-derived part-of-speech tags in Subsection 5.2.2. Third, we consider the application of part-of-speech tags (both human-designed and machine-derived) to novel data, i.e., new alternative sequences in Subsection 5.2.3. Finally, we discuss how we combine the part-of-speech tags with the sequences in Subsection 5.2.4.

5.2.1 The Part-of-Speech Tags

In our experiments we use two types of part-of-speech tags, (1) the human-designed tagset, and (2) a machine-derived tagset.

The first type is the human-designed tagset as included in the British National Corpus. The British National Corpus is tagged with the CLAWS5 tagset. The process of creating the human-designed tagging is described by Leech et al. (1994). The CLAWS5 tagset consists of 57 monolithic tags (see Subsection 5.1.1).

The second type is the machine-derived tagset. We generate the tagset using the methods described by Schütze (1993). The method as employed consists of four phases. They are discussed below.

In the first phase, the neighbourhood vectors are created for each word in the vocabulary. Our vocabulary consists of the 25,000 most frequent words. A neighbourhood vector contains the co-occurrence counts for each word in the vocabulary for a fixed position compared to the word position. We use the co-occurrence positions -2 until 2, i.e., $\langle -2,-1,1,2 \rangle$.

In the second phase, these vectors are clustered. For clustering we use the cosine similarity measure instead of the Euclidian distance measure as is typical. This means that the cosine of the angle between the two vectors is used as the distance metric. For the clustering we employ the CLUTO clustering program (Steinbach et al. 2000).

In the third phase, all clusters are assigned a tag, and the tags are retroactively applied to the corpus. Each word in the corpus is assigned the tag that corresponds to its cluster. We remark that due to this process each word-form gets the same tag, i.e., a word like *walk* will always get the same tag.

In the fourth phase, a machine-learning classifier is used to apply the tags not just to the most frequent 25,000 words, but to all words. In our research, we use a MEMORY-BASED TAGGER model (see the third approach of Subsec-

tion 5.1.4). We apply it to the untagged part of the data (i.e., all tokens not in the 25,000 most frequent words). The result is a fully tagged corpus, with our machine-derived part-of-speech tags.

After employing the four phases of this machine-derived part-of-speech tagging algorithm we end up with two tagged corpora: (1) tagged with the human-designed (CLAWS5) tagset, and (2) tagged with the new machine-derived tagset. Both are used in our experiments.

5.2.2 Evaluation of Machine-Derived Tags

We evaluate the machine-derived tags in three ways, (1) we perform a qualitative analysis, (2) we apply a best-mapping from the machine-derived tags to the human-designed part-of-speech tags, and (3) we calculate the cluster–conditional tag perplexity. The result of our evaluations can be found in Subsection 5.3.1

For the qualitative analysis we examine the clusters of words by hand. Our aim is to achieve some insights into the nature of the tags. We therefore look at the list of some words (say the ten most frequent words) belonging to each cluster.

The best-mapping is created by determining for each machine-derived tag which human-designed tag is the most likely corresponding tag. This is determined using the two tagged version of the BNC (once tagged with the machine-derived tags, once with the human-designed ones), so how often which set of tags co-occurs. The machine-derived tagging is then converted to the best-matching human-designed tag. This conversion is evaluated by using the standard precision, recall, and F-score metrics.

For calculating cluster-conditional tag perplexity we compare the two sets of tags (the human-designed tagset and the machine-derived tagset) as they are applied to the corpus. We use the formula shown in Equation 5.6 to calculate the measure. We also calculate the cluster-conditional tag perplexity values of the human-designed PTB tagset compared to the human-designed BNC tagset. We apply the PTB tagset, which is generated by an MBT-model trained on the PTB. So, we create a frame of reference the reduction in perplexity between two human-designed tagsets, i.e., a baseline value.

5.2.3 Part-of-Speech on New Data

When performing alternative sequence selection a set of alternatives is generated. The elements of this set do not have a part-of-speech tag after generation.

If replacing a word by a different word takes place the part-of-speech tags may change.

Before applying the selection process, the tagging of the alternatives is performed. As in Subsection 5.1.4 (third approach) we use an MBT tagger. The tagger is trained on the same training data as the LME and then applied to the alternative sequences before the LME is applied to them.

5.2.4 Combining Tags and Text

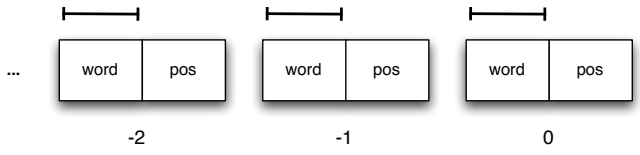
When dealing with sequence classification we base the scores on flexible size n-grams. We want these n-grams to be as large as possible while still providing reliable statistics. In models with local annotation an n-gram can contain part-of-speech tags. In our tagged corpus the part-of-speech tag relevant to a specific token always occurs directly after the token.

We combine tags and text in four different ways. The five ways are listed and explained below. We call them W, PW-wf, PW, P-wf, and W-pwf. We note that the differences between the combination methods are sometimes subtle. When we mention focus position in these descriptions we refer to position w_0 which is the position currently in focus (as in Equation 4.2, reproduced here in Equation 5.7 for convenience).

$$P_n(w_1 \dots w_m) = \prod_{i=1}^{m} P_n(w_i | w_{i-(n-1)}, \dots, w_{i-1}) \tag{5.7}$$

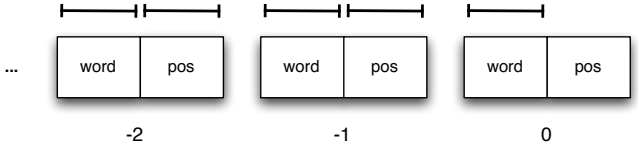
For clarity we provide, for each of the five descriptions, an illustration showing which elements are used at which position (indicated by ⟝⟞). Word tokens are indicated by word, part-of-speech is abbreviated as pos.

W (Word) Our baseline, conforms to the approach described in Chapter 4. It uses only word-tokens at all positions.

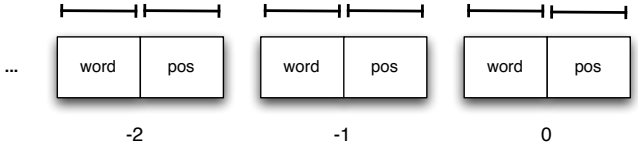


PW-wf (Part-of-speech Word – word focus) Using all part-of-speech tags and all word tokens at all context positions. At the focus position we only

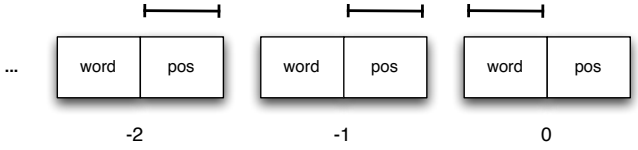
employ the word token and not the part-of-speech tag.



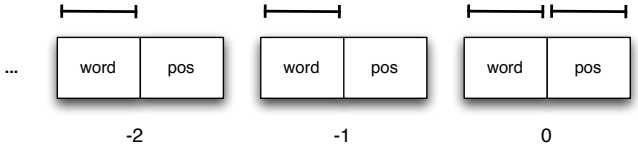
PW (Part-of-speech Word) Using all part-of-speech tags and all word tokens at all positions including the focus position.



P-wf (Part-of-speech – word focus) Using only part-of-speech tags at all positions, only word tokens at the focus position.



W-pwf (Word – part-of-speech word focus) Using only word tokens at all context positions. At the focus position we use both the part-of-speech tags and the word token.



5.3 Experiments

In this section we describe the results of the experiments for the four combination methods PW-wf, PW, P-wf, and W-pwf. We look at the results (1) when using the human-designed tagset, i.e., the gold-standard tagging as present in the BNC, and (2) when using the machine-derived tagset, i.e., the BNC tagged with the tagset derived by the machine according to the methodology outlined in Subsection 5.2.1. The result of the experimental setup is unchanged compared to Chapter 4 Section 4.4.

Before we detail our experimental results in the Subsections 5.3.2 to 5.3.4, we first give an evaluation of the quality of the machine-derived part-of-speech tags in Subsection 5.3.1. We describe our evaluation procedure above in Subsection 5.2.2. To recapitulate we perform three kinds of evaluation: (1) we will perform a qualitative analysis, (2) we will create a best-mapping, apply it, and report on Recall, Precision, and F-score with respect to the gold-standard, and (3) we will report on the cluster conditional tag perplexity of the machine-derived part of speech tagset.

5.3.1 Evaluation of Machine-Derived Part-of-Speech Tags

First we perform a brief qualitative analysis of the assigned clusters. The most frequent tokens for the ten largest clusters are given in Table 5.1. We remark that there is a large disparity in the sizes of the clusters. The smallest cluster consists of 7 types with 18 occurrences, the largest cluster consists of 489 types with 1,223,296 occurrences. We show the differences in cluster occurrences in Figure 5.5. In contrast the differences in the tag occurrences for the human-defined tags is shown in Figure 5.4.

In general the tokens in each cluster seem to be related. This can be clearly seen in Table 5.1. For example, cluster #4 consists partly of past-tense verbs. Other clusters often also show groupings. A second interesting example from Table 5.1 is cluster 8, of which the tokens are less linguistically related. We see a variety of tokens, such as (1) the period, (2) words as in $\{right, anyway, else\}$, and (3) nouns as in $\{money, tea, milk\}$.

Secondly, we create a mapping of each machine-derived tag to a human-designed tag by using the most frequently co-occurring tag. By this tag-to-tag mapping we obtain a corpus that is tagged with the human-designed tagset. We evaluate the tagging by comparing it to the gold-standard tagset. We then calculate an F-score to provide a measure for how well the mapping performs. For the mapping of the machine-derived tags to the gold-standard human-defined tagset of the BNC we achieve the F-score of 0.63. For the human-designed Penn-Treebank tagset to the gold-standard tagset of the BNC achieved the F-score of 0.69. To make the human-designed PTB tags comparable, an MBT tagger was trained on the PTB and applied to the BNC. We sum up the measurements, including the cluster conditional tag perplexity, in Table 5.2. We note that the score achieved by the machine-derived tags is close to the score of the PTB annotation.

Thirdly, we determined the cluster-conditional tag perplexity of the machine-derived tagset compared to the human-designed tagset. As above, we contrast with cluster-conditional tag perplexity of the Penn-Treebank human-designed tagset to the BNC human-designed tagset. The machine-derived tagset has a

tag #	frequent words
1	**,**, that, but, **;**, when, if, then, what, an, ...
2	not, no, only, also, now, such, being, even, still, often, ...
3	of, to, in, for, on, at, by, from, about, into, ...
4	was, had, came, died, suddenly, lay, goes, arrived, fell, lived, ...
5	a, an, very, too, quite, particularly, slightly, completely, somewhat, perfectly, ...
6	and, or, than, both, giving, nor, increasing, finding, showing, keeping, ...
7	are, were, up, **?**, will, may, 're, er, again, ...
8	**.**, right, money, else, trouble, anyway, tea, stuff, milk, mine, ...
9	the, his, which, this, their, my, its, your, our, every, ...
10	i, who, someone, everything, anyone, jesus, somebody, nobody, let's, ...

Table 5.1: A table showing the most frequent words for each of the ten largest clusters found by the machine-derived part-of-speech tagging system.

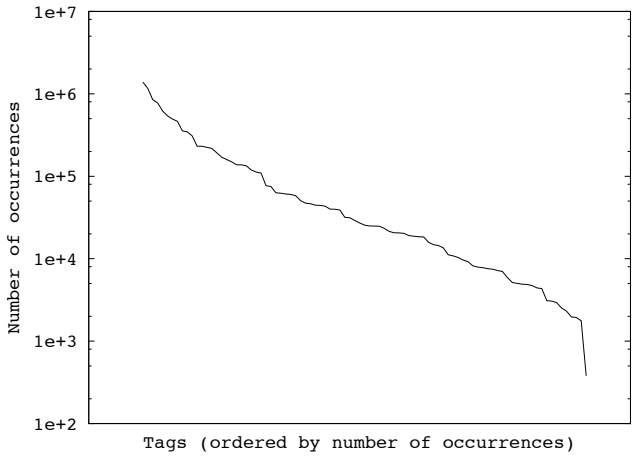


Figure 5.4: A plot showing the number of occurrences of each part-of-speech tag in the human-defined tagging of the BNC. All tags are ordered by number of occurrences and their number of occurrences is shown in the graph. The y-axis is logarithmic.

cluster-conditional tag perplexity of 2.10 on the BNC human-designed tagset. The PTB human-designed tagset has a cluster-conditional tag perplexity of 1.49 on the BNC human-designed tagset. Biemann (2006, 2007) achieved a cluster-conditional tag perplexity with his graph-based method of 2.05 when

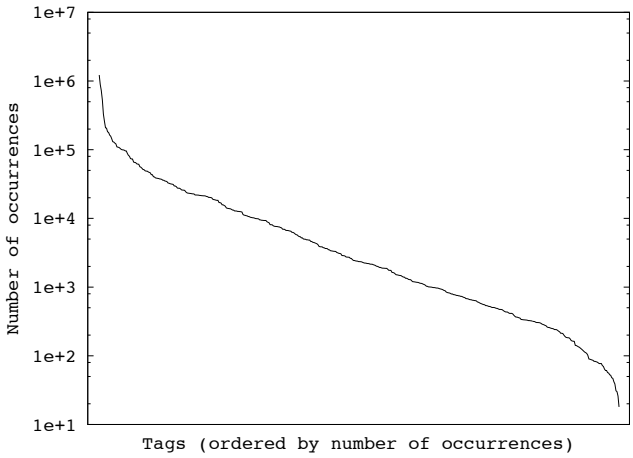


Figure 5.5: A plot showing the number of occurrences of each part-of-speech tag in the machine-derived tagging of the BNC. All tags are ordered by number of occurrences and their number of occurrences is shown in the graph. The y-axis is logarithmic.

	F-score	**CCTP**
machine-derived → gold-standard	0.63	2.10
Penn-TreeBank → gold-standard	0.69	1.49
Biemann (2006)		2.05

Table 5.2: A table showing the results of the evaluations of the machine-derived tagset compared to the human-defined tagset. We show the F-score achieved with the best-mapping and the cluster conditional tag perplexity. Also shown in the cluster-conditional tag perplexity score achieved by Biemann (2006, 2007).

applied to the British National Corpus, i.e., on the same corpus and compared to the same tagset. Our unsupervised clustering is based on the method by Schütze (1993). So, from these observations we may conclude that Schütze's method is still comparable in performance, in this measurement, to a state-of-the-art machine-derived tagging system.

5.3.2 Results on Confusibles

Our series of experiments for the confusibles problem are performed analogously to the series of confusible experiments in Subsection 4.4.1. We run them for two sets of annotations and for five combination methods (W included). For

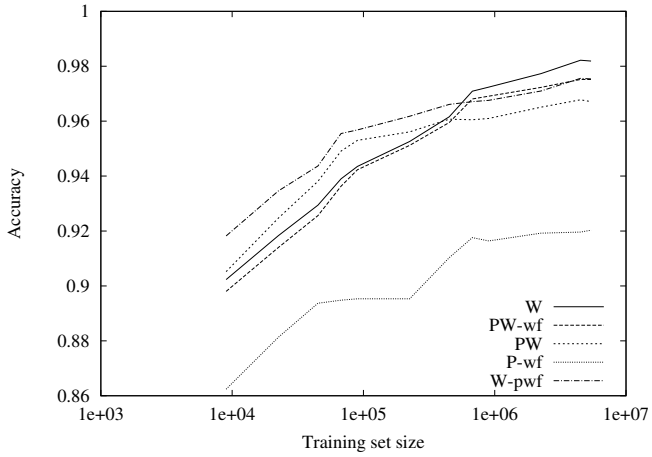


Figure 5.6: Learning curve for the confusible problem using the human-designed tagset. We plot all the 4 combination methods next to the learning curve from Subsection 4.4.1. Y-axis denotes the accuracy (ranging from 0 to 1) and the X-axis the number of sentences used for training. The X-axis is logarithmic.

the two annotations we use (1) the human-defined CLAWS5 tags as present in the BNC and (2) the machine-derived tags on the BNC (see Subsection 5.2.1).

The results are presented as a set of learning curves, (Figure 5.6 shows for the human-designed tags, and Figure 5.7 the machine-derived tags).

We note that, in both cases, the learning curve of P-wf (only tags with the text token as focus) resulted in significantly worse performance, with a p-value $< 2.2e^{-168}$ (between W and P-wf for for the human-designed and the machine-derived part-of-speech tags). P-wf abstracts the contextual information (it only uses part-of-speech tags as context). PW-wf, PW, and W-pwf perform around the level of W, but no better than W. The differences between the different combination methods are significant for the human-designed part-of-speech tags, except between PW-wf and W-pwf (p-value of 0.95 according to McNemar's test, i.e. no significant difference). However, for the full training set the LMEs with added human-designed part-of-speech data perform slightly worse than the LME without (i.e., W).

For the machine-derived tags (see Figure 5.7) the differences between PW-wf, PW, P-wf, and W are not significant, e.g. between W and PW McNemar's test has a p-value of 0.84. For the human-designed tags (see Figure 5.6) we

[8] $< 1.1e^{-16}$ is the smallest value reported by R.

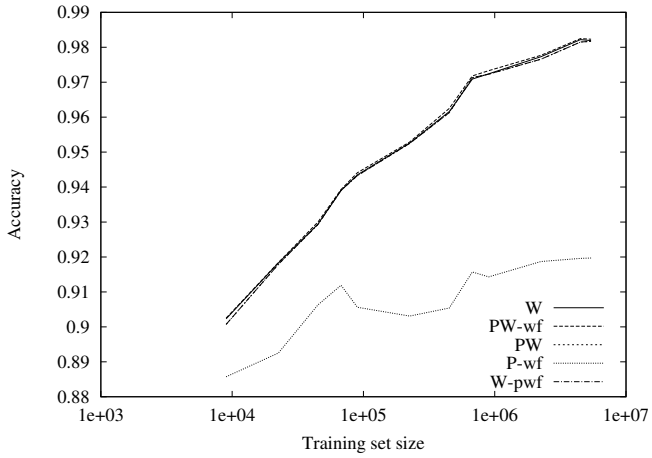


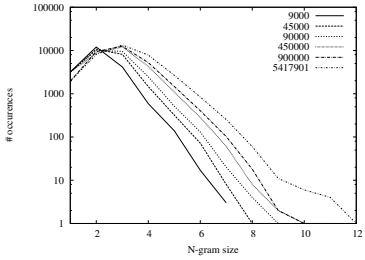
Figure 5.7: Learning curve for the confusible problem using the machine-derived tagset. We plot all the 4 combination methods next to the learning curve from Subsection 4.4.1. Y-axis denotes the accuracy (ranging from 0 to 1) and the X-axis the number of sentences used for training. The X-axis is logarithmic.

see something interesting in the learning curves. That is, we see two methods (PW and W-pwf) clearly[9] outperforming W on small amounts of training data, while at the same time being clearly outperformed when using large amounts of training data[10]. This finding is in accordance with conclusions in the literature, in particular Van den Bosch and Buchholz (2002) already made the argument that part-of-speech information has less value as the training material increases in size. Van den Bosch and Buchholz (2002) also showed that as system without part-of-speech tags will be outperformed by a system using only words, provided that sufficient training material is used.

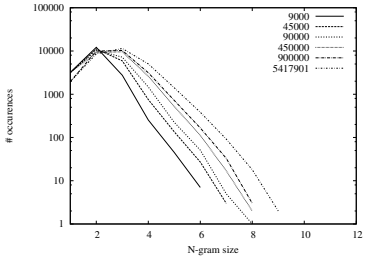
In the Figures 5.8 and 5.9 the sizes of the n-grams used in the classification are shown. The average size of the n-grams used in W, PW-wf, PW and W-pwf is roughly the same. The difference between P-wf compared to the other four methods is most pronounced in the human-designed tagset case (see Figure 5.8(d)), where the average n-gram size used for the full training set is as large as 5.0. For the machine-derived tagset the average n-gram size used by P-wf is 3.9. The reason that the P-wf set has a larger average n-gram size is

[9] For instance, when using 90.000 sentences of training material, between W and W-pwf the p-value is $< 2.2e^{-16}$.

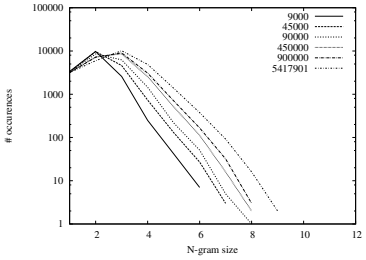
[10] For instance, for the full training set, between W and W-pwf the p-value is $2.2e^{-08}$. So the difference is clearly significant.



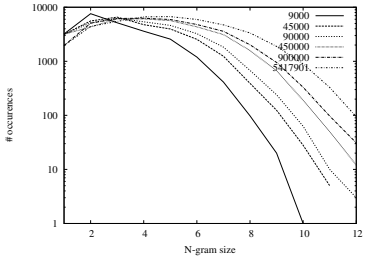
(a) Only text



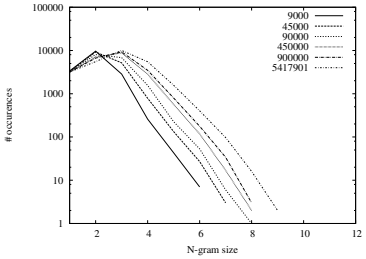
(b) Combination method PW-wf



(c) Combination method PW

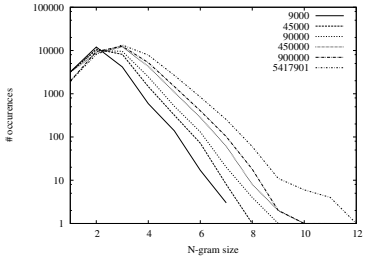


(d) Combination method P-wf

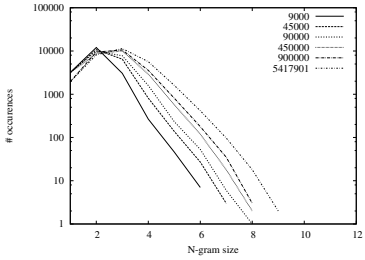


(e) Combination method W-pwf

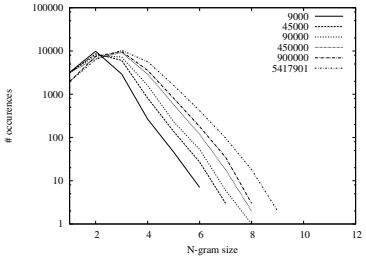
Figure 5.8: Occurrence graph for the confusibles problem on the human-designed tagset showing, for each training set size, the number of times each n-gram size was used in making the alternative selection. The Y-axis is logarithmic. One occurrence graph is shown for each combination method as well as a comparative occurrence graph in (a) for the experiment without any added annotation, as presented in Subsection 4.4.1.



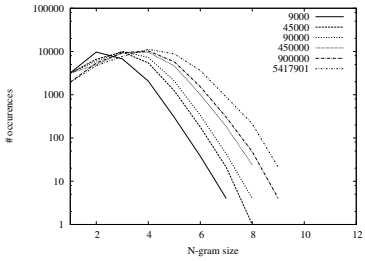
(a) Only text



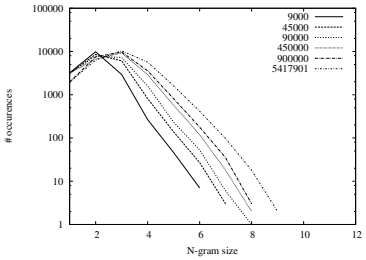
(b) Combination method PW-wf



(c) Combination method PW



(d) Combination method P-wf



(e) Combination method W-pwf

Figure 5.9: Occurrence graph for the confusibles problem on the machine-derived tagset showing, for each training set size, the number of times each n-gram size was used in making the alternative selection. The Y-axis is logarithmic. One occurrence graph is shown for each combination method as well as a comparative occurrence graph in(a) for the experiment without any added annotation, as presented in Subsection 4.4.1.

because part-of-speech tags in itself are less sparse than words, so they match more often. However, as we see in the accuracy results discussed above, having a larger context does not aid performance if an alternative with smaller context has sufficiently rich data, e.g., as in the part-of-speech tag context versus the word context.

5.3.3 Results on Verb and Noun Agreement

Our series of experiments for the verb and noun agreement problem are performed analogously to the series of the verb and noun agreement experiments in Subsection 4.4.2. As for the confusible experiments (see Subsection 5.3.2) we run them for two sets of annotations and for five combination methods (W included). The results of the learning curves are shown in Figure 5.10 for the human-designed tags, and in Figure 5.11 for the machine-derived tags. We also visualize the size of the n-grams used for classification in a series of occurrence graphs. This series can be seen in Figure 5.12 for the human-designed tagset and in Figure 5.13 for the machine-derived tagset.

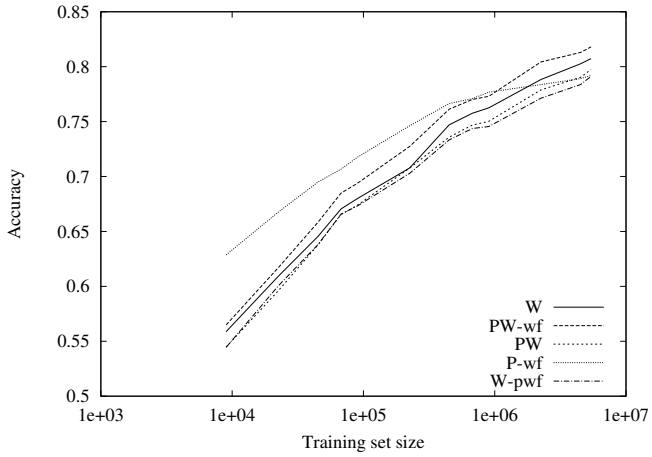


Figure 5.10: Learning curve for the verb and noun agreement problem using the human-designed tagset. We plot all the 4 combination methods next to the original learning curve from Subsection 4.4.2. Y-axis denotes the accuracy (from 0 to 1) and the X-axis the number of sentences used for training. The X-axis is logarithmic.

We note that all the learning curves perform roughly the same, with a few exceptions. One exception occurs in case of the machine-derived tagset, where

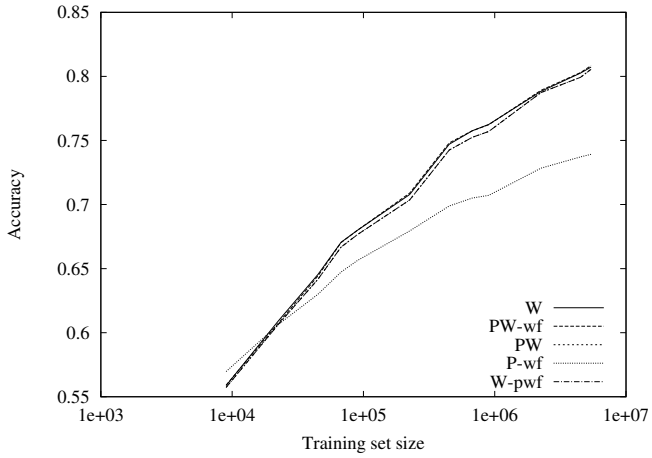
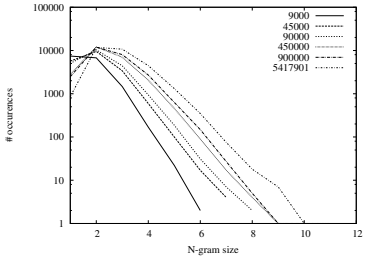


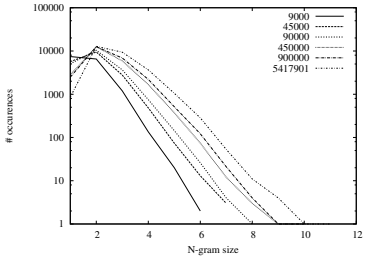
Figure 5.11: Learning curve for the verb and noun agreement problem using the machine-derived tagset. We plot all the 4 combination methods next to the original learning curve from Subsection 4.4.2. Y-axis denotes the accuracy (from 0 to 1) and the X-axis the number of sentences used for training. The X-axis is logarithmic.

the combination method P-wf is clearly and significantly outperformed by the other methods, as with the experiments on the confusibles in Subsection 5.3.2. This is remarkable as PW-wf and P-wf significantly outperform the text-only curve in case of the human-designed tagset[11]! In fact, even with the full-size training set PW-wf significantly outperforms the baseline W (p-value of $8.2e^{-06}$). From this we deduce that, in case of verb and noun agreement, the human-designed part-of-speech tags provide extra relevant information that the machine-derived tagset does not. This extra information is only beneficial to the results with the full-size training data in cases where it compliments the wordform information.

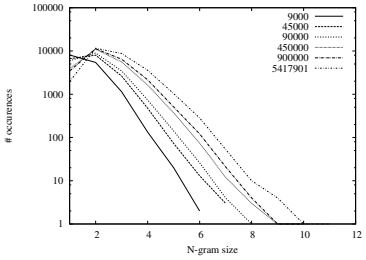
[11] At 450.000 sentences of training material, the p-value for W versus PW-wf is $4.8e^{-09}$. At 450.000 sentences of training material, for W versus P-wf the p-value is $8.2e^{-05}$.



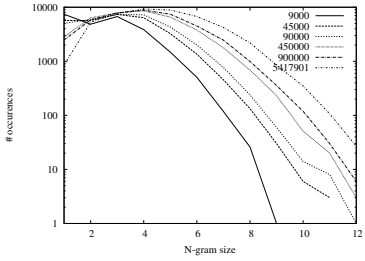
(a) Only text



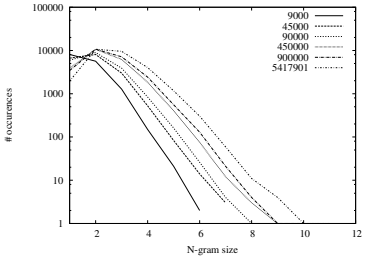
(b) Combination method PW-wf



(c) Combination method PW

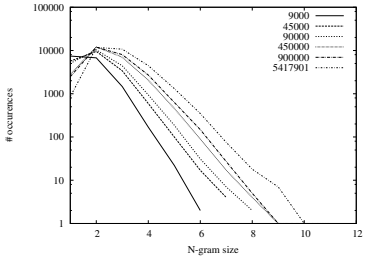


(d) Combination method P-wf

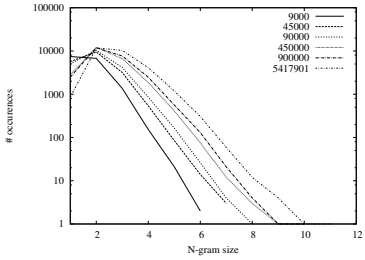


(e) Combination method W-pwf

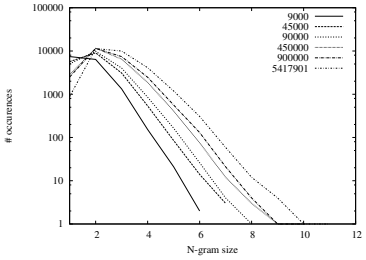
Figure 5.12: Occurrence graph for the verb and noun agreement problem on the human-designed tagset showing, for each training set size, the number of times each n-gram size was used in making the alternative selection. The Y-axis is logarithmic. One occurrence graph is shown for each combination method as well as a comparative occurrence graph in (a) for the original experiment in Subsection 4.4.2.



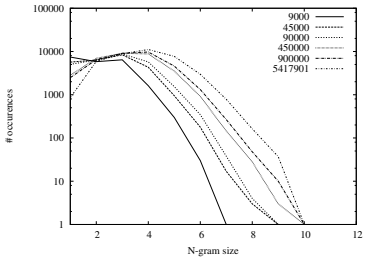
(a) Only text



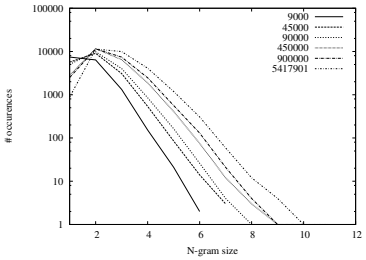
(b) Combination method PW-wf



(c) Combination method PW



(d) Combination method W-pwf



(e) Combination method P-wf

Figure 5.13: Occurrence graph for the verb and noun agreement problem on the machine-derived tagset showing, for each training set size, the number of times each n-gram size was used in making the alternative selection. The Y-axis is logarithmic. One occurrence graph is shown for each combination method as well as a comparative occurrence graph in (a) for the original experiment in Subsection 4.4.2.

5.3.4 Results on Prenominal Adjective Ordering

Our series of experiments for the prenominal adjective reordering problem is, just like the series of experiments for the confusibles problem and the verb and noun agreement problem, performed analogously to the series in the previous chapter. These learning curves are shown in Figure 5.14 for the learning curves using the human-designed tags, and in Figure 5.15 for the learning curves using the machine-derived tags. In the Figures 5.16 and 5.17 we visualise the size of the sizes of the n-grams used for classification in a series of occurrence graphs.

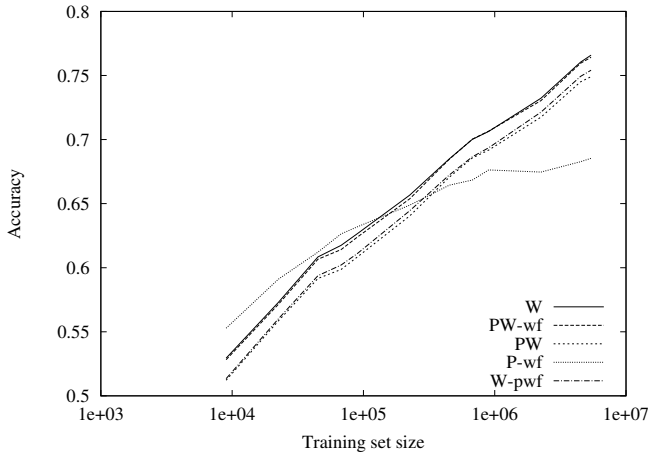


Figure 5.14: Learning curve for the prenominal adjective reordering problem using the human-designed tagset. We plot all the 4 combination methods next to the original learning curve from Subsection 4.4.3. Y-axis denotes the accuracy (from 0 to 1) and the X-axis the number of sentences used for training. The X-axis is logarithmic.

We note that the learning curves all perform roughly similar. In the case of the human-defined tagset there is one exception in the form of the P-wf learning curve. The other combination methods perform roughly the same, but their small difference in performance is significant with one exception. The exception is in the case of PW-wf, were the difference with W is not significant, with a p-value of 0.26 according to McNemar's test. The P-wf learning curve initially performs the best, but with the full-size training it is clearly (p-value $< 2.2e^{-16}$) outperformed by the rest. The context of the P-wf curve consists of only the part-of-speech tags, out of our combination methods it is the only one without words in the context. The results suggest that for the prenominal adjectives re-ordering problem the larger context generated by this method is

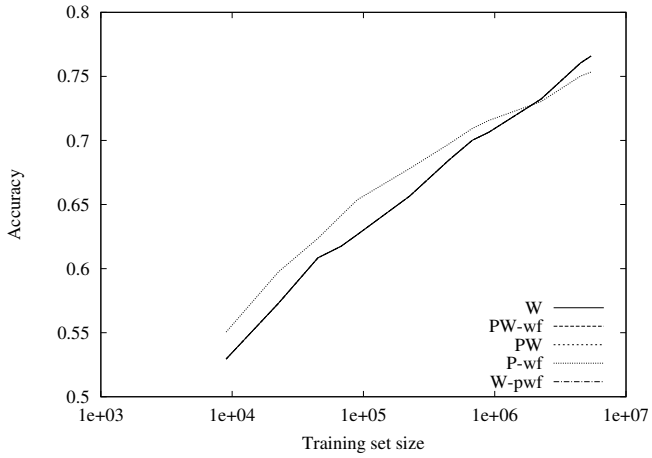
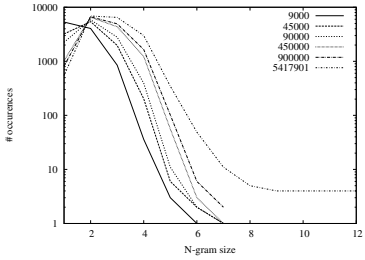


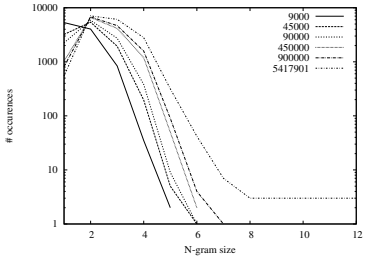
Figure 5.15: Learning curve for the prenominal adjective reordering problem using the machine-derived tagset. We plot all the 4 combination methods next to the original learning curve from Subsection 4.4.3. Y-axis denotes the accuracy (from 0 to 1) and the X-axis the number of sentences used for training. The X-axis is logarithmic.

initially more important, but later the simplified information in that context is detrimental compared to the full text information.

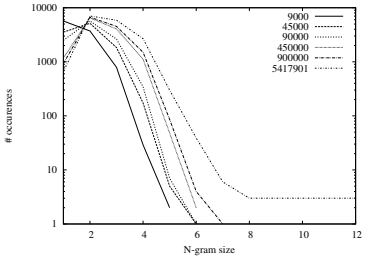
For the learning curves of the machine-derived tagging experiments we see a similar behaviour, i.e., we see P-wf initially outperforming the other combination methods, while P-wf is outperformed with larger amounts of training material (significantly with a p-value of $9.7e^{-3}$). All other combination methods, i.e., W, PW-wf (p-value of 1), PW (p-value of 1), and W-pwf (p-value of 1) do not significantly differ in performance in these experiments. We did not observe this behaviour of P-wf for the machine-derived tagging experiments in the results for the other two alternative sequence selection tasks. Therefore, we may conclude that, at least with small amounts of training material, the larger context provided by the P-wf method is more important to the performance of the prenominal adjective reordering task.



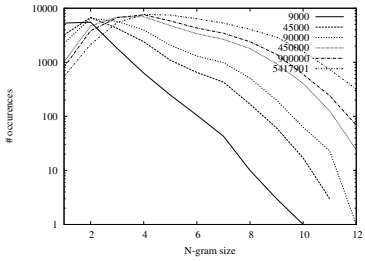
(a) Only text



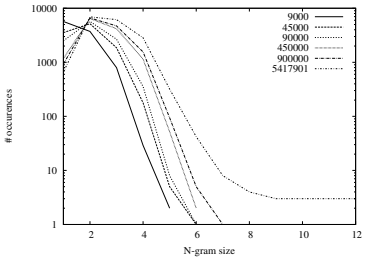
(b) Combination method PW-wf



(c) Combination method PW

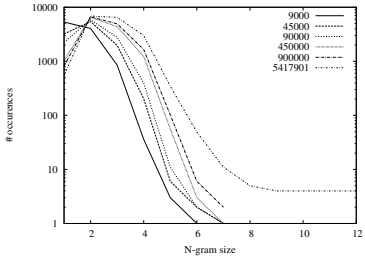


(d) Combination method P-wf

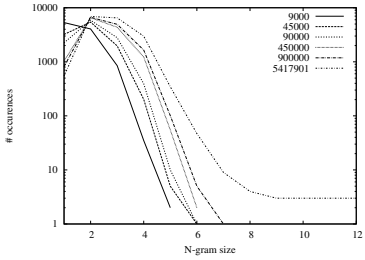


(e) Combination method W-pwf

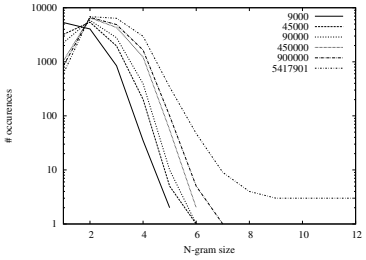
Figure 5.16: Occurrence graph for the prenominal adjective reordering problem on the human-designed tagset showing, for each training set size, the number of times each n-gram size was used in making the alternative selection. The Y-axis is logarithmic. One occurrence graph is shown for each combination method as well as a comparative occurrence graph in (a) for the original experiment in Subsection 4.4.3.



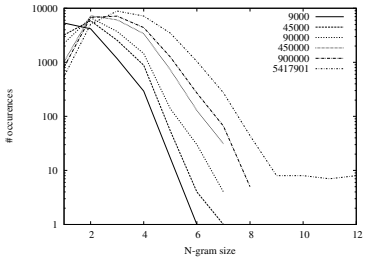
(a) Only text



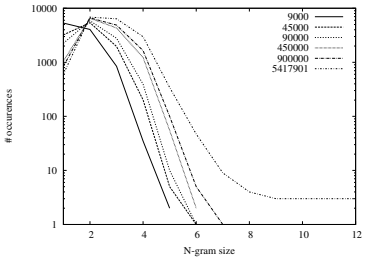
(b) Combination method PW-wf



(c) Combination method PW



(d) Combination method P-wf



(e) Combination method W-pwf

Figure 5.17: Occurrence graph for the prenominal adjective reordering problem on the machine-derived tagset showing, for each training set size, the number of times each n-gram size was used in making the alternative selection. The Y-axis is logarithmic. One occurrence graph is shown for each combination method as well as a comparative occurrence graph in (a) for the original experiment in Subsection 4.4.2.

5.4 Partial Answers to RQ3 and RQ4

In this chapter we investigated a language-model environment with local annotation that uses ∞-grams and synchronous back-off. The investigations provide a basis for achieving partial answers for the research questions RQ3 and RQ4. We repeat RQ3 and RQ4 below. They are followed by our partial answers.

> **Research Question 3:** Is there a benefit to including annotations in the language model, measured as a better performance on alternative sequence selection tasks?

We attempt to answer RQ3 employing three observations. First, given sufficient training material little benefit is gained by including locally dependent annotation, which in our investigation are based on part-of-speech tags. However, we found two exceptions mentioned here as our second and third observations. Second, in the case of verb and noun agreement, the presence of the human-designed part-of-speech tags adds useful information, resulting in a significantly higher performance. Third, in all three alternative sequence selection tasks the presence of the human-designed tagset improves the performance when a limited amount of training data is available. So, for small amounts of training data there is an advantage.

> **Research Question 4:** Is there a difference in performance on alternative sequence selection tasks when using human-designed annotations compared to machine-generated annotations?

We attempt to answer RQ4 by employing two observations. First, we see from the results above that the machine-derived part-of-speech tags generally perform similarly compared to a system without annotation. Second, we observe that for the experiments using the human-designed part-of-speech tags there is an improvement in performance for small amounts of training data. These answers raise the question as to why the addition of machine-derived part-of-speech tags does not result in performance improvement.

We note that the system with the human-defined annotation provides novel information to the system, which we can observe in the results being different compared to the system without annotation. In contrast, the system with the machine-derived annotation performs nearly identical to the system without annotation. We may speculate that the human-designed part-of-speech tags, being designed and defined by humans, actually *add* information. In contrast, the machine-derived tags do *not add* new information to the corpus. It is even worse, when using the full training set, the system without annotation outperforms, or performs identical to the experiments with annotations. The answer to

RQ4, is that human-designed annotation provides an advantage over machine-derived annotation, only for small training sets.

5.5 Chapter Conclusions

In this chapter we investigated a language-model environment that can be characterized as a model with local annotation. We focussed on the effect of adding such annotation, and on the difference in performance between the human-designed part-of-speech tagset and the machine-derived part-of-speech tagset.

The experiments performed on the three alternative selection problems led to detailed results for the language-model environment described in this chapter. The result of the two types of annotation (human-designed and machine-derived) were compared to each other, as well as to the results from Chapter 4. From the experiments with combinations of annotations (using five methods) we may conclude the following. (1) The machine-derived part-of-speech tagset does not add any useful new information to the model. (2) The human-designed part-of-speech tagset does add useful information to the model. (3) As the size of the training material increases the addition of the part-of-speech tags becomes less and less influential on the performance of the system (when compared to the system without annotations), up to the point where its addition reduces to zero. As we observed in Section 5.3 this point occurs around one million sentences of training material for all three alternative sequence selection tasks.

Chapter 6

Models with Complex Annotation

This chapter describes an LME that is to be characterised as a model with complex annotation. We focus on the effect of adding dependency information to the language-model environment. The LME described in this chapter also includes the aspects discussed in Chapter 4, i.e., the ∞-gram and synchronous back-off. There are many types of complex annotations such as complex annotations such as constituency or dependency parses, semantic role graphs, or anaphoric link structures. In this chapter we focus on adding information derived from dependency parses. We choose dependency structures as they are a relatively straightforward and well-studied type of syntactic annotation. Now we are ready to examine RQ3 and RQ4 with respect to the complex annotation model for the three alternative sequence selection problems. Below we re-iterate both RQs.

> **Research Question 3:** Is there a benefit to including annotations in the language model, measured as a better performance on alternative sequence selection tasks?

> **Research Question 4:** Is there a difference in performance on alternative sequence selection tasks when using human-designed annotations compared to machine-generated annotations?

The course of this chapter reads as follows. In Section 6.1 we describe two computational approaches to dependency parsing.

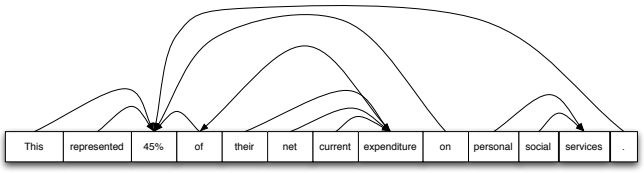


Figure 6.1: Example dependency parse made by the Maltparser on the BNC. Here we only show the dependency relations.

In Section 6.2 we complement the language-model environment as described in Chapter 3 with a precise description of (1) the use of human-designed dependency parses, and (2) the generation and use of machine-derived dependency parses. This section builds on the information given in Section 4.3. That information still applies.

In Section 6.3 we describe our experiments on the three alternative sequence selection problems for the experiments with added dependency information. The results of the experiments are analysed with respect to RQ3 and RQ4.

In Section 6.4 we explicitly formulate our answers to RQ3 and RQ4 for the models with complex annotation. In Section 6.5 we provide a summary and give our conclusions.

6.1 Dependency Parses

A well-known complex annotation type is the dependency parse. We call dependency parses complex because rather than being associated with individual tokens, dependencies represent cross-token information that may span any distance within the sentence. Dependency parses denote the lexical dependencies within the sequence (Jurafsky and Martin 2000). We illustrate the concept in Figure 6.1. For each word in the sequence a relation is assigned to a headword of that word, i.e., the headword is the word on which the word depends syntactically.

Dependency parses are a popular type of parses. Usually, supervised machine-learning systems are employed to assign parses to new, previously unseen, sentences. Dependency parsing has been the subject of several CoNLL shared tasks. The first CoNLL shared task on dependency parsing was in 2006 (Buchholz and Marsi 2006), it continued in 2007 (Nivre et al. 2007a), 2008 (Surdeanu et al. 2008), and 2009 (Hajič et al. 2009). Dependency parsers often use the presence of a part-of-speech tagging in order to improve tractability.

Several approaches to supervised dependency parsing have been tried, such as spanning-tree based algorithms (McDonald and Pereira 2006, McDonald et al. 2005), projective dependency parsing (Nivre 2003, Nivre et al. 2007b), memory-based learning (Nivre et al. 2004, Morante et al. 2009a,b), and constraint satisfaction (Canisius and Tjong Kim Sang 2007).

Below, we will briefly discuss (1) supervised dependency parsing as designed by human experts, and (2) unsupervised dependency parsing based on statistics as derived by a computer.

6.1.1 Supervised Dependency Parsing

Here, we briefly discuss some approaches to supervised dependency parsing. The shared aspect of these approaches is that they are all based on graph theory. The main difference is in the decision scope. Decisions are made for each word in the input sequence to decide which other word is its headword. One major difference between the two approached mentioned above is using the entire search space as decision context versus using the local part of the search space to make a greedy decision.

McDonald et al. (2005) introduced a well-known supervised dependency parser called MSTPARSER. The dependency parsing problem is reduced to a maximum-spanning-tree search problem. A spanning tree is a graph-theoretic concept that denotes a tree over a graph that includes all vertices, but no cycles. Chu and Liu (1965) provides an insightful description of this concept. A maximum spanning tree is the spanning tree over a graph of which the combined edge value is the largest possible value for a spanning tree on the graph in question. Edge values are based on observations made on the training corpus, i.e., an often observed relation will have a larger value than an infrequently observed one. The parser that the MSTParser generates makes decisions based on the entire search space in an exhaustive manner. It examines all possible dependency parses.

Nivre et al. (2006) introduced another well-known supervised dependency parser, the MALTPARSER (see also Hall et al. 2007). Nivre et al. (2006) describe the MALTPARSER as a data-driven parser. The parser that the MALTPARSER generates makes decisions online and in a greedy manner. (Here online means that it processes the sequence token by token and makes a decision at each step.) Due to its focus on local token-level decisions, the classifier in MALTPARSER can be enriched with a large feature space.

In two publications, McDonald and Nivre (2007) and Nivre and McDonald (2008) make a direct comparison between the MSTPARSER and the MALTPARSER. They mainly compare the *locality* and *nature* of the errors made by

the parsers. This is contrasted to the feature-poor but exhaustive approach of the MSTPARSER on the one side and the feature-rich and greedy approach of the MALTPARSER on the other side.

A third approach is proposed by Canisius (2009). They propose to use constraint satisfaction inference to tackle dependency parsing. Three classifiers generate possible constraints, based on tokens and pairs of tokens. These constraints are combined using weighted constrained satisfaction (maximising the value of the selected constraints) resulting in a dependency parse of the sequence. For the combination step the CYK algorithm (Younger 1967, Eisner 2000) is used.

6.1.2 Unsupervised Dependency Parsing

We briefly discuss a selection of approaches to unsupervised dependency parsing We start by describing the approach by Gorla et al. (2007) which is closely related to the MST (maximum spanning tree) approach. We will describe several other approaches as well.

Gorla et al. (2007) describe an unsupervised dependency parsing method that uses a modified mutual information formula on sequences of part of speech tags. They modify the MI formula with a distance component and use it to generate an MST over possible dependency parses, much like McDonald and Nivre (2007) did on supervised data. The weights of all the possible connections in the graph are based on co-occurence observations in the training data.

Spitkovsky et al. (2009) describe a scaffolding approach, where the system is first developed using sentences of length one. At each step the resulting model of the previous step is used to initiate training. In the first step, sentences with length one, the element in the sentence is sure to attach to the root-position. In the second step, sentences with length two, there is a single choice to make, i.e., which of the two elements attaches to the other, where the other will then attach to the root position. The decision on the second step are made with the help of statistics from the first step and afterwards all statistics are updated. These steps are repeated until the entire training set is processed.

Seginer (2007) describes an incremental dependency parser that is learned by an efficient unsupervised learning algorithm. The common cover link (CCL) parser works directly on the raw natural language data without depending on annotations such as part-of-speech tags.

The CCL-system creates an incremental parser which can be used to parse new sentences. Incrementality means that the parser reads a sentence word by word and for each new word it looks in the history, to add, if possible, some dependency and parse structure. For each incremental step the parser looks at

all possible links it can add, adds the highest-scoring link out of those possible links and then repeats the process. Adding a link alters the set of possible links that can be added to the sentence at hand. For this process the parser depends on the use of a lexicon with adjacency points. Adjacency points are a collection of the most frequent co-occuring words, with their counts and their relative position (to the left or to the right). At each step of the learning process (i.e., every time a step of one word is taken) the lexicon is updated.

Unsupervised dependency parsing systems have been used successfully in language modelling for Japanese Kana-Kanji conversion, realising a 3.5% reduction in error-rate compared to the same system without the unsupervised dependency information. Gao and Suzuki (2003) described a system where they incorporated unsupervised dependency relations into a language model. Their approach to the incorporation of such information is similar to that of Chelba and Jelinek (1998), who combined parts of supervised parses in an n-gram language model.

6.2 Language-Model Environment

The LME employed in this chapter is based closely on the LME of Chapter 5. However, instead of annotating with part-of-speech tags we use a human-designed and a machine-derived dependency-parse structure to add information to the model. The information generated by these two systems is added in the form of tags. As the tag value the corresponding headword of the word in focus is added.

We remark that there is an important distinction between the human-defined dependency parses and the machine-derived dependency parses. The human-defined dependency parses also contain several types of information that we do not use, such as the part-of-speech tags of words, and the types of dependency relations between words.

Below we will examine the structure found by the two different dependency parsers. We will investigate the average distance from each word to its corresponding headword for each dependency parser, as well as the overall structure of the parses.

6.3 Experiments

In this section we describe the results of the experiments for this chapter. The experiments are structured similarly to Section 5.3. As before we will examine

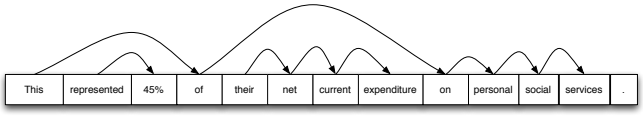


Figure 6.2: Example parse made by the CCLparser on the BNC.

the performance of the LME using the five methods (see Subsection 5.2.4) on the three alternative sequence selection tasks for the two annotation methods. These annotations are (1) the human-designed dependency parse, and (2) the machine-derived dependency parse. The performance achieved by the language model is reported in Subsections 6.3.2 to 6.3.4.

In Subsection 6.3.1 we start by examining some aspects of the machine-derived dependency parses. We will focus on two aspects: (1) the distance between each word and its attached headword, and (2) the overall structure of the parses. These aspects are compared to their analogues in the human-designed dependency parses.

6.3.1 Comparing Dependency Parses

In this subsection we give quantitative and qualitative comparisons between the two parsing systems. The MALTPARSER provides the human-defined dependency parse structure and the CCLPARSER provides the machine-derived dependency parse structure.

First, we look at the distance of each word to its corresponding headword on the BNC. We observe that the average distance from the current word to its headword is rather different. In case of the MALTPARSER the average distance from the focus word to its headword is 3.2 word positions on the original sentence. In case of the CCLPARSER the average distance from the focus word to its headword is 1.4 word positions. This difference is explained by the inherent bias of the CCLPARSER to establish local dependencies.

Secondly, we look at the overall structure of the parses. We do this with two telling examples, one for each parser. The example for the MALTPARSER is given in Figure 6.1. The example for the CCLPARSER is given in Figure 6.2. Both figures show an example sentence, with each line-and-arrow connecting a word with its corresponding headword. The examples illustrate that the CCLPARSER discovers mainly local relations.

These observations imply that with human-designed dependency annotations, many words link to a headword at a longer distance. In case of the machine-

derived dependency annotation most words link to the next word as headword, at least on English data. With the way we add the information to the model, i.e., using the linked headword as a tag, the former seems more useful.

6.3.2 Results on Confusibles

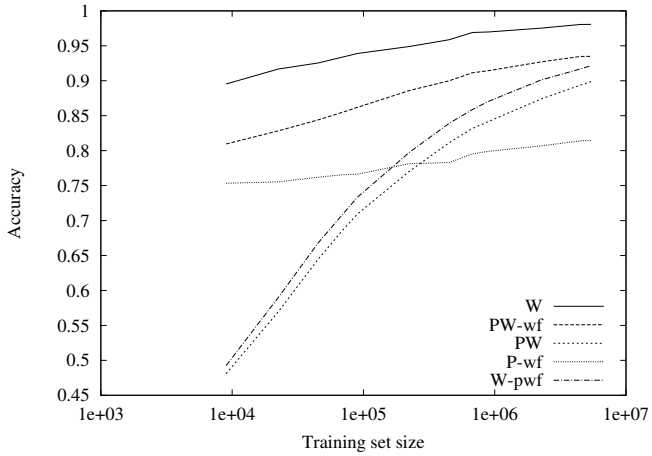


Figure 6.3: Learning curve for the human-defined dependency parses. We plot all the five methods (baseline included). Y-axis denotes the accuracy (from 0 to 1) and the X-axis the number of sentences used for training. The X-axis is logarithmic.

The execution of our series of experiments for the confusibles problem is analogous to the series of experiments in Subsection 4.4.1 and Subsection 5.3.2. We calculate the learning curves for the five different combination methods (baseline included). Figure 6.3 shows the results for the human-designed dependency parses and Figure 6.4 shows the results for the machine-derived dependency parses.

The sizes of the n-grams used for the selections are measured. We visualize the size of the sizes of the n-grams used for classification in a series of occurrence graphs. The average size of the n-grams used is rather similar (see Figure 6.5 and 6.6).

We observe that, in both cases, the learning curve of the combination method P-wf (only tags with the text token as focus) resulted in the worst performance. P-wf only has the headword information of the context to base its decisions on, and not the words themselves. Consequently, the most important contextual

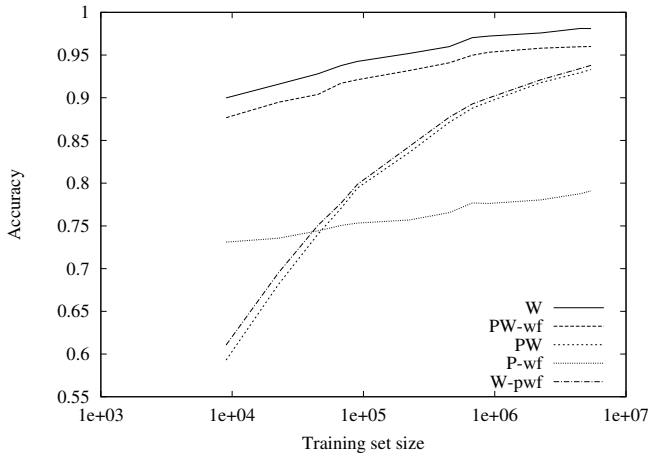
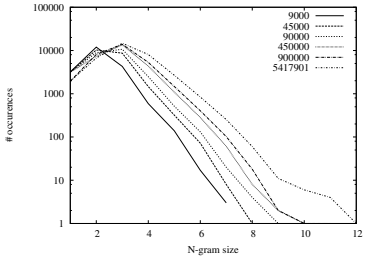


Figure 6.4: Learning curve for the machine-derived dependency parses. We plot all the five methods (baseline included). Y-axis denotes the accuracy (from 0 to 1) and the X-axis the number of sentences used for training. The X-axis is logarithmic.

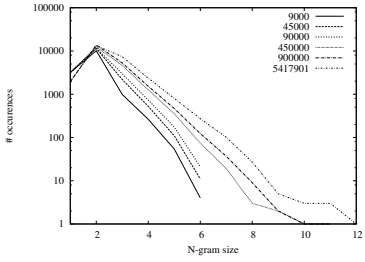
information is not used. Furthermore we observe that, unlike P-wf, all other methods approach the performance of the model without added annotations as the training set size increases.

For both the human-defined and the machine-generated annotation we observe that the differences between PW-wf, PW, W-pwf, and W decrease as the training size increases. We observe that the human-defined annotation approaches the learning curve without annotation less than the machine-derived annotation. For both cases we observe that at no point the enriched learning curves outperform the original curve W. The difference of the learning curves with annotation to the original curve W is significant in all cases with a p-value $< 2.2e^{-16}$ for both the human-designed and the machine-derived annotations.

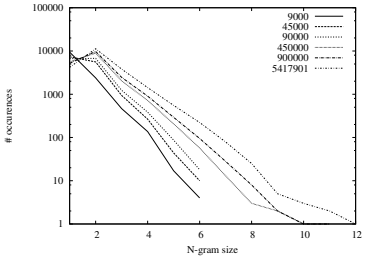
We also observe that, for both annotations, the curves for PW and W-pwf start with a performance much lower than would be expected when compared to the other learning curves. If we observe the amount of items for which the system cannot make a decision we see that for those two curves the system encounters too much sparseness to make a decision at the start of the learning curve. Note that if the system does not make a decision, the item is counted as erroneous. We show the results of these observations only for the human-designed dependency parse in Figure 6.7. We postulate that this is caused by too much sparseness, even when just using the focus position. When PW and W-pwd only use the focus position they try to calculate the probability $P(wordtag)$.



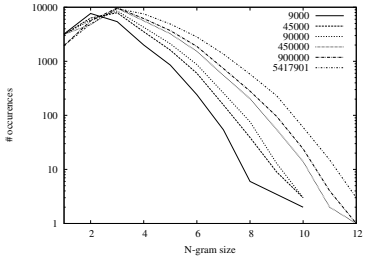
(a) Only text



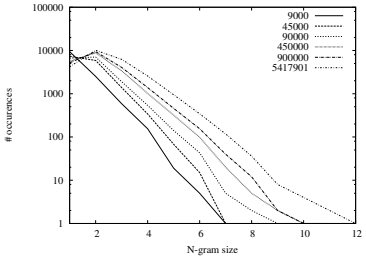
(b) Combination method PW-wf



(c) Combination method PW

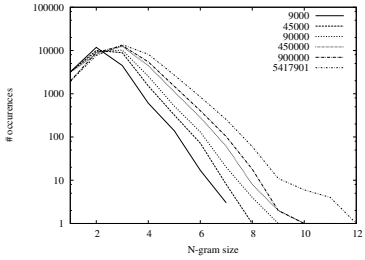


(d) Combination method P-wf

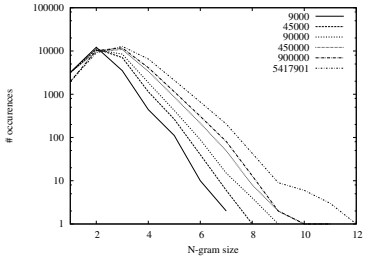


(e) Combination method W-pwf

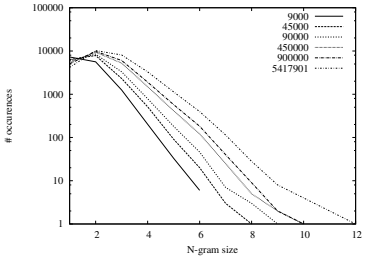
Figure 6.5: Occurrence graph for the confusibles problem on the human-defined dependency parses. For each training set size, the number of times each n-gram size was used in making the alternative selection is shown. The Y-axis is logarithmic. One occurrence graph is shown for each combination method as well as a comparative occurrence graph in (a) for the original experiment in Subsection 4.4.1.



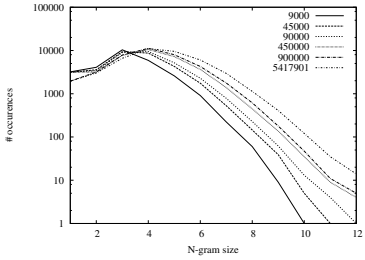
(a) Only text



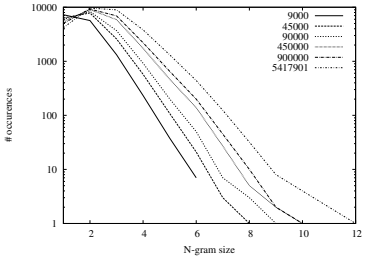
(b) Combination method PW-wf



(c) Combination method PW



(d) Combination method P-wf



(e) Combination method W-pwf

Figure 6.6: Occurrence graph for the confusibles problem on the machine-derived dependency parses. For each training set size, the number of times each n-gram size was used in making the alternative selection is shown. The Y-axis is logarithmic. One occurrence graph is shown for each combination method as well as a comparative occurrence graph in (a) for the original experiment in Subsection 4.4.1.

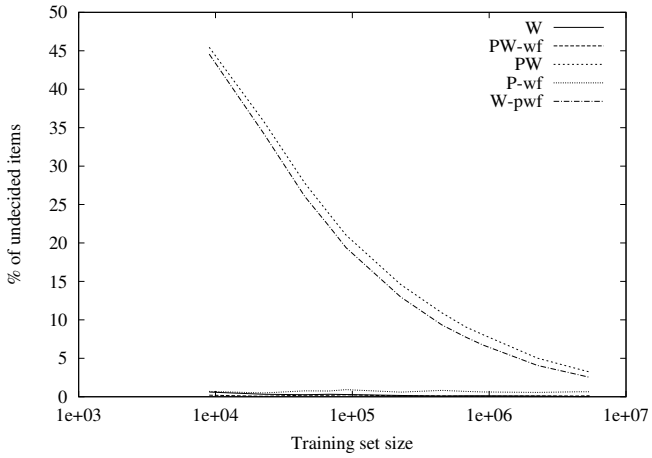


Figure 6.7: Learning curve for the confusible problem using the human-defined tagset. The curve shows the percentage of items for which the system was unable to make a decision. The X-axis denotes the number of sentences used for training. The X-axis is logarithmic.

As the tag is also a word in the case of dependency parsing, it causes the language model to be too sparse to make a decision, especially for small amounts of training data.

6.3.3 Results on Verb and Noun Agreement

The execution of our series of experiments for the verb and noun agreement problem was as follows. Analogously to Subsection 4.4.2 and Subsection 5.3.3 we run a series of experiments to create learning curves for all four combination methods involving annotations. We give the results as a series of graphs. In Figure 6.8 we show the learning curves for the human-designed dependency parses and in Figure 6.9 we show the learning curves for the machine-generated dependency parses.

We observe that, similarly to the results for the confusibles in Subsection 6.3.2, the systems with annotations do not outperform the system from Chapter 4 at any point. Second, similarity with the results for the confusibles is that the combination methods PW and W-pwf again perform quite badly with small amounts of training data. This is caused by the same reason as in the confusibles experiments from Subsection 6.3.2. The difference of the learning

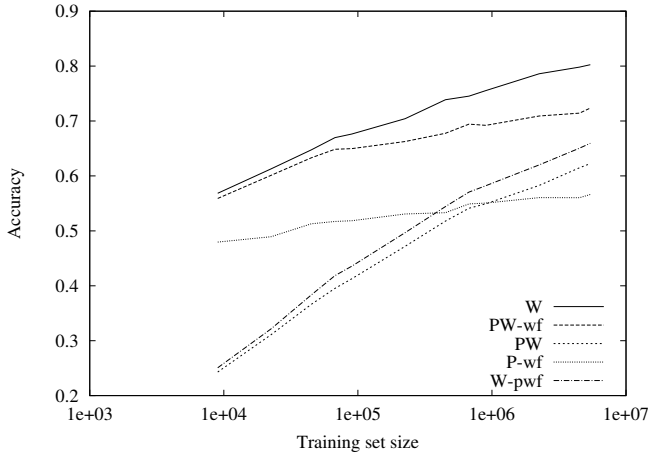


Figure 6.8: Learning curve for the human-defined dependency parses on the verb and noun agreement problem. We plot all the five methods (baseline included). Y-axis denotes the accuracy (from 0 to 1) and the X-axis the number of sentences used for training. The X-axis is logarithmic.

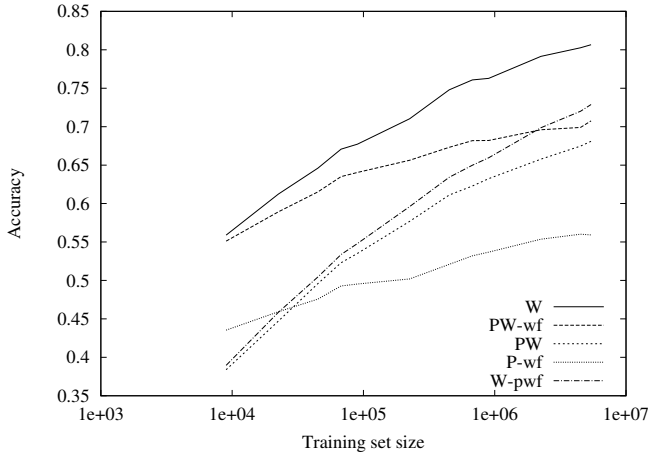
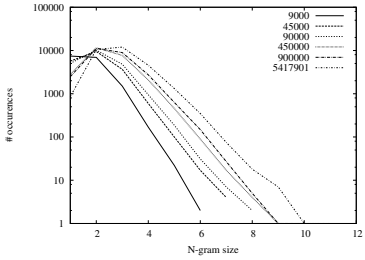
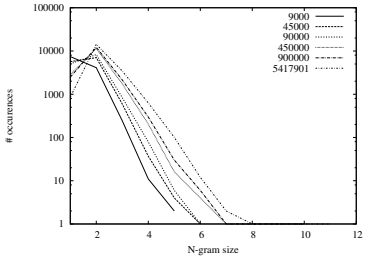


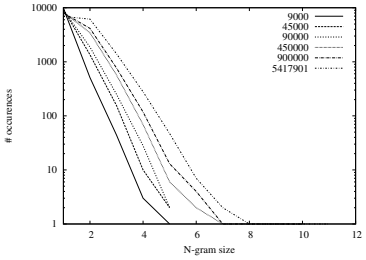
Figure 6.9: Learning curve for the machine-derived dependency parses on the verb and noun agreement problem. We plot all the five methods (baseline included). Y-axis denotes the accuracy (from 0 to 1) and the X-axis the number of sentences used for training. The X-axis is logarithmic.



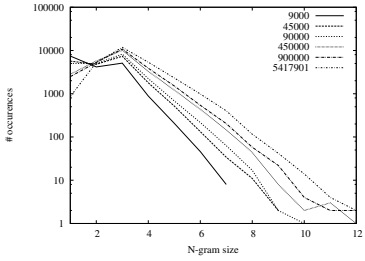
(a) Only text



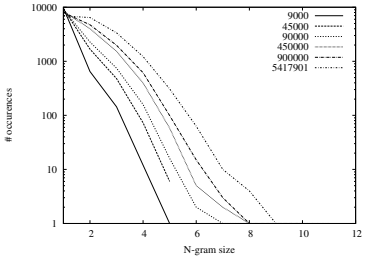
(b) Combination method PW-wf



(c) Combination method PW

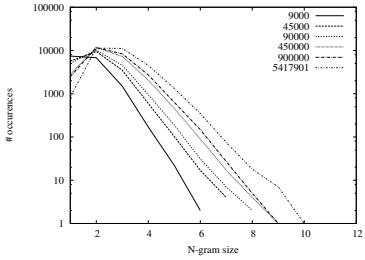


(d) Combination method P-wf

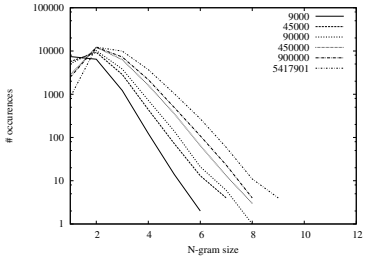


(e) Combination method W-pwf

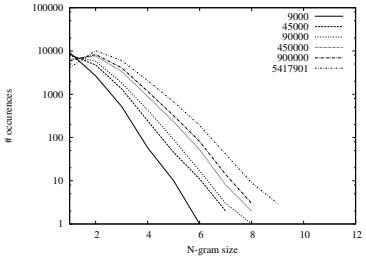
Figure 6.10: Occurrence graph for the verb and noun agreement problem on the human-defined dependency parses. For each training set size, the number of times each n-gram size was used in making the alternative selection is shown. The Y-axis is logarithmic. One occurrence graph is shown for each combination method as well as a comparative occurrence graph in (a) for the original experiment in Subsection 4.4.2.



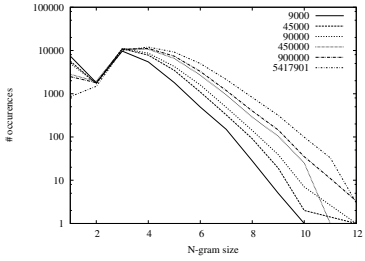
(a) Only text



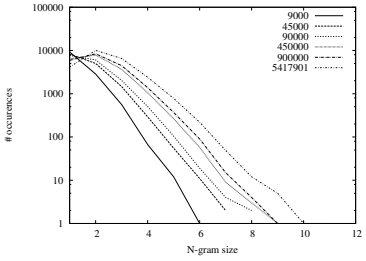
(b) Combination method PW-wf



(c) Combination method PW



(d) Combination method P-wf



(e) Combination method W-pwf

Figure 6.11: Occurrence graph for the verb and noun agreement problem on the machine-derived dependency parses. For each training set size, the number of times each n-gram size was used in making the alternative selection is shown. The Y-axis is logarithmic. One occurrence graph is shown for each combination method as well as a comparative occurrence graph in (a) for the original experiment in Subsection 4.4.2.

curves with annotation to the original curve W is significant in all cases with a p-value $< 2.2e^{-16}$ for both the human-designed and the machine-derived annotations.

6.3.4 Results on Prenominal Adjective Ordering

The execution of our series of experiments for the prenominal adjective reordering problem is as follows. We run the experiments analogously to the experiments on prenominal adjective reordering in Subsection 4.4.3 and Subsection 5.3.4 The resulting learning curves are shown in Figure 6.12 for the human-defined dependency parses and in Figure 6.13 for the machine-derived dependency parses. Furthermore, in Figure 6.14 we show the occurrence graphs for the human-designed dependency annotations and in Figure 6.15 the corresponding occurence graphs for the machine-generated dependency annotations.

We note that the learning curves all perform roughly similar. The exceptions here are the methods PW and W-pwf for both annotations. We remark that the similarity between PW and W-pwf is the use of both the text-token and the annotation on the focus position. This opposed to W, PW-wf, and P-wf which all only use the text-token at the focus position. This observation suggests that, with a text token and annotation token in the focus position the language model suffers from increased sparseness. As with the confusibles and the verb and noun agreement problem earlier in the chapter we observe that this is due to the system not being able to make a decision due to the sparseness. For both the human-defined and machine-derived dependency annotation the difference of the curve for W with the curves with annotation is significant with a p-value $< 2.2e^{-16}$.

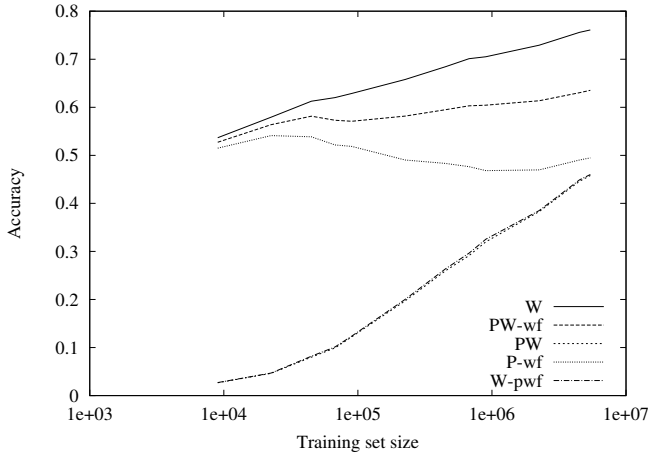


Figure 6.12: Learning curve for the human-defined dependency parses on the prenominal adjective ordering problem. We plot all the five methods (baseline included). Y-axis denotes the accuracy (from 0 to 1) and the X-axis the number of sentences used for training. The X-axis is logarithmic.

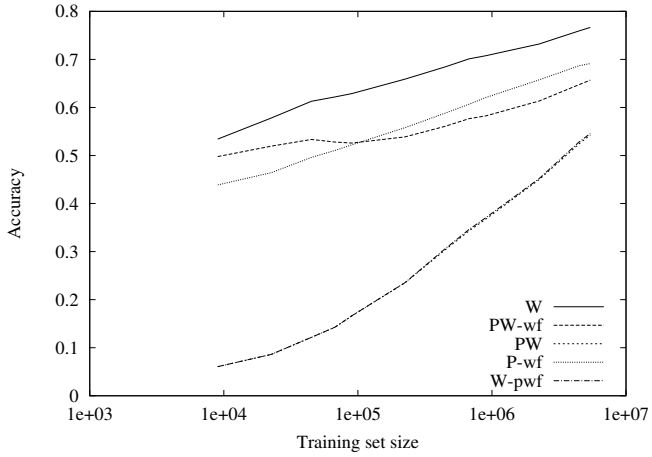
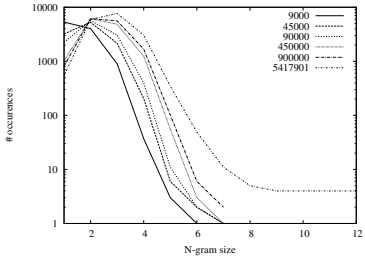
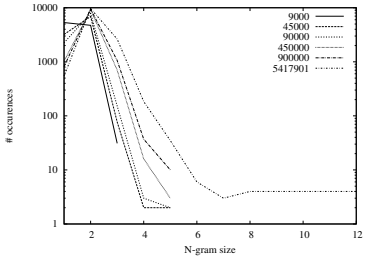


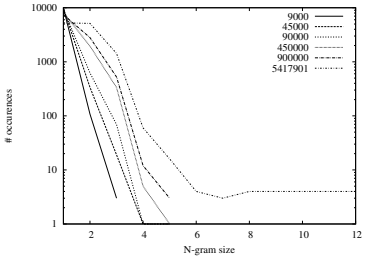
Figure 6.13: Learning curve for the machine-derived dependency parses on the prenominal adjective ordering problem. We plot all the five methods (baseline included). Y-axis denotes the accuracy (from 0 to 1) and the X-axis the number of sentences used for training. The X-axis is logarithmic.



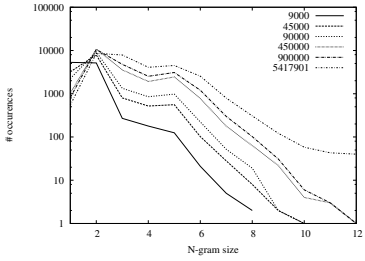
(a) Only text



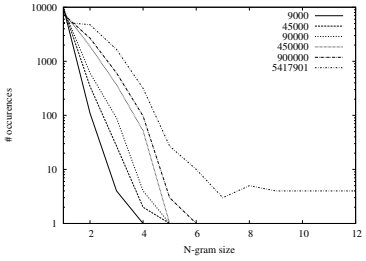
(b) Combination method PW-wf



(c) Combination method PW

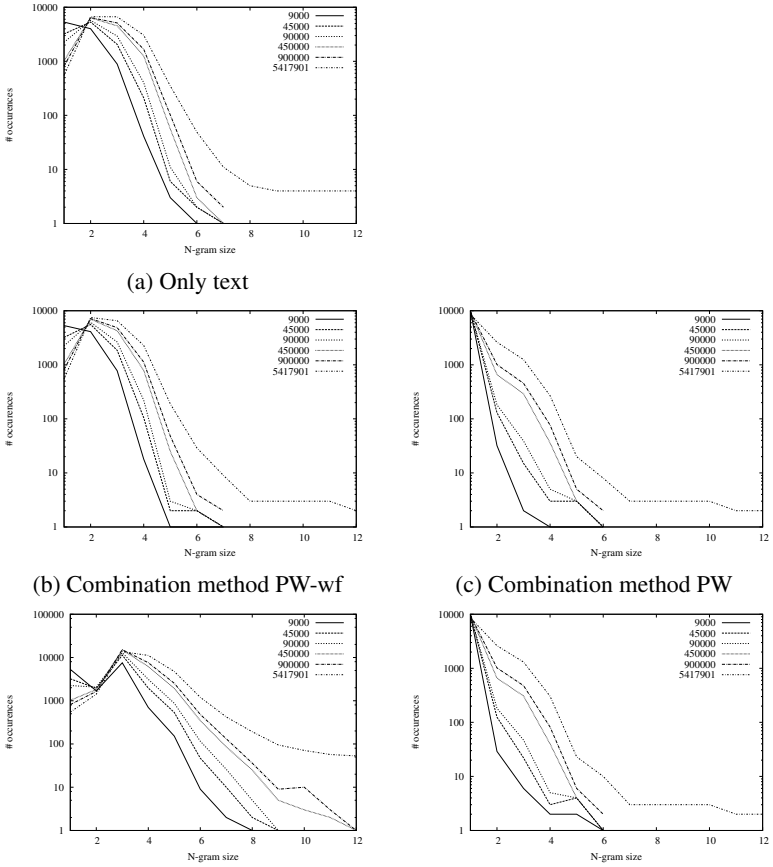


(d) Combination method P-wf



(e) Combination method W-pwf

Figure 6.14: Occurrence graph for the prenominal adjective ordering problem on the human-defined dependency parses. For each training set size, the number of times each n-gram size was used in making the alternative selection is shown. The Y-axis is logarithmic. One occurrence graph is shown for each combination method as well as a comparative occurrence graph in (a) for the original experiment in Subsection 4.4.3.



(a) Only text

(b) Combination method PW-wf

(c) Combination method PW

(d) Combination method P-wf

(e) Combination method W-pwf

Figure 6.15: Occurrence graph for the prenominal adjective ordering problem on the machine-derived dependency parses. For each training set size, the number of times each n-gram size was used in making the alternative selection is shown. The Y-axis is logarithmic. One occurrence graph is shown for each combination method as well as a comparative occurrence graph in (a) for the original experiment in Subsection 4.4.3.

6.4 Partial Answers to RQ3 and RQ4

In this chapter we investigated a language-model environment with dependency annotation that uses ∞-grams. The investigations provide a basis to achieve partial answers for the research questions RQ3 and RQ4. We repeat RQ3 and RQ4 below. They are followed by our partial answers.

> **Research Question 3:** Is there a benefit to including annotations in the language model, measured as a better performance on alternative sequence selection tasks?

We attempt to answer RQ3 by employing two observations made on all the results presented above. First, no benefit is gained by including dependency annotation in the manner as presented above. Second, we observe that the methods with text-token and annotation token in the focus position (i.e., PW and W-pwf) perform much worse than the other methods due to the sparseness problems. Near the full training set they all converge to the results from Chapter 4 (the system without annotation). However, all methods that use annotation perform significantly worse than the method without annotation (i.e., W). In summary, this type of annotation does not contribute to a better performance.

> **Research Question 4:** Is there a difference in performance on alternative sequence selection tasks when using human-designed annotations compared to machine-generated annotations?

We attempt to answer RQ4 by employing two observations on all the results presented above. We see in the results above that the language models for both types of dependency annotations perform worse than the language model without annotation. The differences between the two annotations is most apparent when looking at them qualitatively. We observe that the human-defined annotation uses much longer dependencies than the machine-derived dependencies. Furthermore, in case of the human-defined annotation there is additional information available that we do not use (as the machine-derived dependencies lack this information).

The results presented in this chapter raise the question as to why the addition of dependency annotation does not result in an increase of performance. We may speculate that including the headwords directly results in an increase in sparseness, instead of combatting sparseness. Here, we remark that the approach of Gao and Suzuki (2003) also directly uses the headword. In all cases the system without annotation outperforms the experiments with annotations. In brief, human-designed dependency annotation does not provide an advantage over machine-generated dependency annotation in the way it is used here.

6.5 Chapter Conclusions

In this chapter we investigated a language-model environment that can be characterised as a model with dependency annotation. We focussed on the effect of adding such annotations, and on the difference in performance between the human-designed dependency annotation and the machine-derived dependency annotation.

The experiments performed on the three alternative sequence selection problems led to detailed results for the language-model environment described in this chapter. The result of the two types of annotation (human-defined and machine-derived) were compared to each other, as well as to the results of Chapter 4. From the experiments with combinations of annotations we may conclude the following: (1) dependency annotations, when employed in this manner, hamper the performance of the model; and (2) when using a merged token combining the annotation and the word token in the focus position the model performs considerable worse, due to sparseness.

However, human-designed dependency annotations usually contain extra information that we do not employ. In the case of the human-designed dependency annotation as used in this chapter this extra information includes the type of dependency relation and part-of-speech tags.

Chapter 7

Conclusions and Future Work

This chapter provides our final conclusions on language models for alternative sequence selection and recommendations for future work. For clarity we provide an overview of the best results from all language model systems (from Chapters 4, 5, and 6) in Table 7.1. The conclusions will be given in two parts. First, in Section 7.1 we give our conclusions for each research question in turn. Second, in Section 7.2 we answer our problem statements by using the answers to our research questions.

In addition to the conclusions, in Section 7.3 we will give recommendations for future work.

7.1 Answering the Research Questions

We will commence by, once again, reiterating our research questions. Each RQ will be followed by a brief conclusion for the domain of alternative sequence selection.

> **Research Question 1:** Is there a need to predetermine or limit the size of the n-grams used in language models? Is there an inherent advantage or disadvantage to using a fixed-size n?

In the general case we are facing an ambiguous answer: restricting the size of the n-gram is a disadvantage, however it does not impair performance. The

LM type			confusibles	verb and noun	adjectives
no annotation			**98.19%**	80.73%	**76.59%**
part-of-speech	human-defined	PW	96.71%	79.74%	74.91%
		P-wf	92.03%	79.20%	68.54%
		W-pwf	97.54%	79.12%	75.41%
		PW-wf	97.52%	**81.80%**	**76.42%**
	machine-generated	PW	**98.17%**	80.57%	**76.59%**
		P-wf	91.97%	73.92%	75.34%
		W-pwf	**98.17%**	80.55%	**76.59%**
		PW-wf	**98.23%**	80.84%	**76.59%**
dependency	human-defined	PW	89.90%	62.25%	45.79%
		P-wf	81.44%	56.63%	49.48%
		W-pwf	92.17%	65.95%	46.02%
		PW-wf	93.50%	72.36%	63.53%
	machine-generated	PW	93.30%	68.10%	54.27%
		P-wf	79.11%	55.92%	69.13%
		W-pwf	93.80%	72.88%	54.58%
		PW-wf	96.00%	70.75%	65.68%

Table 7.1: An overview of the results achieved by all LMEs (from Chapters 4 5, and 6). Results are given in percentage of correct decisions (accuracy). Numbers in **bold face** are the best results in each column and do not differ significantly (McNemar test p-value > 0.05). They do differ significantly with the numbers not in bold face.

natural limit for the n-gram is reached quite soon in practice, and this natural limit is determined by the largest size n-gram. It is a disadvantage to fix the n in cases where a probability for a larger subsequence than n tokens.

> **Research Question 2:** If the size of the n-grams is not fixed in advance, how can we still generate comparable distributions when we select among alternative sequences?

Owing to synchronous back-off we were able to show that there is no need to predetermine and restrict the size of the n-gram in advance. However, the sparseness of the training data used will naturally limit the size of the n-gram that synchronous back-off uses for decisions. The optimal size of the n-gram used for a decision between alternative sequences can differ for each set of alternative sequences. This leads us to conclude that there is an advantage in not predetermining the size of the n-gram. Using synchronous back-off, we generate comparable distributions by synchronising the back-off mechanism over alternative sequences. From our observations in Section 4.2 it is clear that synchronous back-off works better than the other language model strategies discussed in that section.

> **Research Question 3:** Is there a benefit to including annotations in the language model, measured as a better performance on alternative sequence selection tasks?

We attempt to answer RQ3 on the basis of three observations. We observe that, given sufficient training material little benefit is gained by including locally dependent annotations such as part-of-speech tags. In our experiments, we found two exceptions. First, in the case of verb and noun agreement, the presence of human-designed part-of-speech tags adds useful information, resulting in a significantly higher performance. Second, in all three alternative sequence selection tasks the presence of the human-designed tagset significantly improved the performance when using a small training set. This does not hold for the machine-derived tagset where there is no improvement and no significant difference between the different systems. So it appears that, only for small amounts of training data there exists an advantage when employing human-designed part-of-speech tags.

In contrast, we face the following two observations relating to the dependency annotations. First, no benefit is gained by including dependency annotation in the manner proposed here. Second, we observe that the methods PW and W-pw perform much worse than the other combination methods due to the sparseness problems. In summary, dependency annotation does not deliver any benefit and so it does not contribute to a better performance. The most likely reason is sparseness due to directly including the headword as annotation.

> **Research Question 4:** Is there a difference in performance on alternative sequence selection tasks when using human-designed annotations compared to machine-generated annotations?

We attempt to answer RQ4 with two observations. First, we see in the results of Chapter 5 that the machine-generated part-of-speech tags generally perform similarly (not significantly different) as the experiments without annotation. The exception to this is P-wf, in which case it performs significantly worse than the experiments without annotation. Second, we observe that for the experiments with the human-designed part-of-speech tags, there is a significant difference in performance between the experiments without annotation versus the experiments with annotations.

However, for the dependency annotations (Chapter 6), we see in the results that both dependency annotations perform significantly worse than the experiments without annotation.

In brief, we may conclude that using human-defined part-of-speech annotations can be useful for small amounts of training data.

In our experiments we observe that the machine-generated annotations do not aid in the performance of the LME. In contrast, the human-defined annotations aid performance in several, specific, cases. We speculate that this is the case because the machine-derived annotations model information that is already present in the data itself.

7.2 Answering the Problem Statements

Here we will reiterate our problem statements, followed by the answers as they pertain to each problem statement. We will formulate our answers in terms of the answers to the RQs.

> **Problem Statement 1 (Flexibility):** Is it helpful to create a statistical language model that is flexible, i.e., not fixed in advance, with regards to the n-gram size, for adequately handling the problem of sparseness?

In Chapter 4 we answered RQ1 and RQ2. We showed that it is possible to determine flexibly the size of the n-gram on a case-by-case basis. In theory this is highly desirable, as it results in a model that better approximates the ideal language model. In practise, the synchronous back-off method performs rather better on alternative sequence selection than the other back-off methods

in our preliminary experiments, as shown in Section 4.2. However, we observe that the average n-gram size used tends to be around 3. We also observed that almost no n-grams of a size greater than 5 were used. In brief, we have created a flexible language model that automatically determines the optimal n-gram size to use.

> **Problem Statement 2 (Annotations):** Do linguistically motivated annotations and their automatically generated counterparts provide information that can be successfully used as a back-off step to handle sparseness? Does alleviating sparseness in this way increase performance on alternative sequence selection tasks?

In the Chapters 5 and 6 we answered RQ3 and RQ4, determining the answer to this problem statement. In those chapters we observed, and concluded, that the addition of linguistically motivated annotations as well as automatically derived annotations do not provide the language model system with additional useful information except in quite specific cases. Those specific cases involve the human-designed part-of-speech tags and limited amounts of training material. Should the amount of training material be further increased, then the importance of the additions will continue to decrease.

7.3 Recommendations and Future Work

We complete this chapter by five recommendations for possible avenues of further study.

First, for building language models, it holds that investigating the effects of using annotations on the performance can be beneficial if annotations are already available. However, using more unannotated data is a more reliable source for improving results.

Second, further studies on the effects of dependency annotations should be performed. For instance, by not using the headword as a main feature, but by using aspects pertaining to it as additional features (e.g., the part-of-speech tag of the headword, the type of dependency relation). This could reduce sparseness and might provide more usable long-distance information.

Thirth, fourth, and fifth, two related concepts could be further studied in the context of our language models. The third recommendation is to encourage the creation of more flexible, automatically discovered, back-off paths (i.e., not user defined as in Bilmes and Kirchhof (2003) or this thesis), and the fourth recommendation is the investigation of flexibly implementing traditional smoothing and interpolation methods in our language models. Furthermore, we note

that it is limiting in many cases to only use the left context for calculating the probability of a word in a sequence, as well as using a fixed back-off path. So, the fifth recommendation is to investigate more general models of language that include both left and right context when available, and flexible back-off strategies. For instance, Van den Bosch (2006b) uses both left and right context in a word prediction system based on k-nearest neighbour classification.

References

Abouelhoda, M., Kurtz, S., and Ohlebusch, E. (2004). Replacing suffix trees with enhanced suffix arrays. *Journal of Discrete Algorithms*, 2(1):53–86.

Adriaans, P. and Van Zaanen, M. (2004). Computational grammar induction for linguists. *Grammars*, 7:57–68. Special issue with the theme "Grammar Induction".

Baase, S. and Gelder, A. V. (2000). *Computer algorithms*. Addison Wesley Longman, Boston.

Baayen, R. H., Piepenbrock, R., and van Rijn, H. (1993). *The CELEX lexical data base on CD-ROM*. Linguistic Data Consortium, Philadelphia, PA.

Bailey, T. and Jain, A. K. (1978). A note on distance-weighted k-nearest neighbor rules. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-8(4):311–313.

Banko, M. and Brill, E. (2001). Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 26–33. Association for Computational Linguistics.

Biemann, C. (2006). Unsupervised part-of-speech tagging employing efficient graph clustering. In *Proceedings of the 21st International Conference on computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, COLING ACL '06, pages 7–12, Stroudsburg, PA, USA. Association for Computational Linguistics.

Biemann, C. (2007). *Unsupervised and Knowledge-free Natural Language Processing in the Structure Discovery Paradigm*. PhD thesis, Leipzig University.

Bilmes, J. and Kirchhof, K. (2003). Factored language models and generalized parallel backoff. In *Proceedings of HLT/NACCL, 2003*.

Brants, T. (2000). TnT – a statistical part-of-speech tagger. In *Proceedings of the 6th Applied NLP Conference, ANLP-2000, April 29 – May 3, 2000, Seattle, WA*, pages 224–231.

Brill, E. (1992). A simple rule-based part-of-speech tagger. In *Proceedings of the Third ACL Applied NLP*, pages 152–155, Trento, Italy.

Buchholz, S. and Marsi, E. (2006). CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL-X, the Tenth Conference on Computational Natural Language Learning*, New York, NY.

Busser, G. and Morante, R. (2005). Designing an active learning based system for corpus annotation. In *Proceedings of the XXI Congresso de la Sociedad Espanola para el Procesamiento del Lenguaje Natural, SEPLN-2005*, pages 375–381, Granada, Spain.

Canisius, S. (2009). *Structured prediction for natural language processing: A constraint satisfaction approach*. PhD thesis, Tilburg University.

Canisius, S. and Tjong Kim Sang, E. (2007). A constraint satisfaction approach to dependency parsing. In *Proc. of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1124–1128, Prague, Czech Republic.

Canisius, S. and Van den Bosch, A. (2004). A memory-based shallow parser for spoken Dutch. In Decadt, B., De Pauw, G., and Hoste, V., editors, *Selected papers from the Thirteenth Computational Linguistics in the Netherlands Meeting*, pages 31–45. University of Antwerp.

Canisius, S. and Van den Bosch, A. (2007). Recompiling a knowledge-based dependency parser into memory. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP-2007)*, pages 104–108, Borovets, Bulgaria.

Charniak, E., Carroll, G., Adcock, J., Cassandra, A., Gotoh, Y., Catz, J., Littman, M., and McCann, J. (1996). Taggers for parsers. *Artificial Intelligence*, 85:45–57.

Chelba, C. and Jelinek, F. (1998). Exploiting syntactic structure for language modeling. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Montréal, Quebec, Canada*.

Chen, S. and Goodman, J. (1996). An empirical study of smoothing techniques for language modelling. In *Proceedings of the 34th Annual Meeting of the ACL*, pages 310–318. ACL.

Chen, S. and Goodman, J. (1998). An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University.

Chodorow, M. and Leacock, C. (2000). An unsupervised method for detecting grammatical errors. In *Proceedings of NAACL'00*, pages 140–147.

Chu, Y. and Liu, T. (1965). On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

Church, K. and Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Cost, S. and Salzberg, S. (1993). A weighted nearest neighbour algorithm for learning with symbolic features. *Machine Learning*, 10:57–78.

Cover, T. M. and Hart, P. E. (1967). Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, 13:21–27.

Cutting, D., Kupiec, J., Pedersen, J., and Sibun, P. (1992). A practical Part-of-Speech tagger. In *Proceedings Third ACL Applied NLP*, pages 133–140, Trento, Italy.

Daelemans, W., Van den Bosch, A., and Zavrel, J. (1999). Forgetting exceptions is harmful in language learning. *Machine Learning, Special issue on Natural Language Learning*, 34:11–41.

Daelemans, W., Zavrel, J., and Berck, P. (1996a). Part-of-speech tagging for Dutch with MBT, a memory-based tagger generator. In Van der Meer, K., editor, *Informatiewetenschap 1996, Wetenschappelijke bijdrage aan de Vierde Interdisciplinaire Onderzoeksconferentie Informatiewetenchap*, pages 33–40, The Netherlands. TU Delft.

Daelemans, W., Zavrel, J., Berck, P., and Gillis, S. (1996b). MBT: A memory-based part of speech tagger generator. In Ejerhed, E. and Dagan, I., editors, *Proceedings of the Fourth Workshop on Very Large Corpora*, pages 14–27. ACL SIGDAT.

Daelemans, W., Zavrel, J., Van der Sloot, K., and Van den Bosch, A. (2010). TiMBL: Tilburg memory based learner, version 6.3, reference guide. Technical Report ILK 10-01, ILK Research Group, Tilburg University.

De Jong, F. (1983a). Numerals as determiners. In Bennis, H. and Van Lessen Kloeke, W., editors, *Linguistics in the Netherlands*, pages 95–104, Dordrecht, The Netherlands. Foris.

De Jong, F. (1983b). Sommige niet, andere wel; de verklaring van een raadselachtig verschil. In *GLOT 6*, pages 229–246, Dordrecht, The Netherlands.

de la Higuera, C. (2005). A bibliographical study of grammatical inference. *Pattern Recognition*, 38:1332–1348.

de la Higuera, C. (2010). *Grammatical Inference, Learning Automata and Grammars*. Cambridge University Press, Cambridge.

Devijver, P. A. and Kittler, J. (1982). *Pattern recognition. A statistical approach*. Prentice-Hall, London, UK.

Dudani, S. (1976). The distance-weighted k-nearest neighbor rule. In *IEEE Transactions on Systems, Man, and Cybernetics*, volume SMC-6, pages 325–327.

Eisner, J. (2000). Bilexical grammars and their cubic-time parsing algorithms. In Bunt, H. and Nijholt, A., editors, *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62. Kluwer Academic Publishers, Norwell, MA, USA.

Even-Zohar, Y. and Roth, D. (2000). A classification approach to word prediction. In *Proceedings of the First North-American Conference on Computational Linguistics*, pages 124–131, New Brunswick, NJ. ACL.

Feist, J. (2008). *The order of premodifiers in English nominal phrases*. PhD thesis, The University of Auckland, Auckland, NZ.

Francis, W. and Kučera, H. (1982). *Frequency Analysis of English Usage*. Houghton Mifflin Company, Boston, MA.

Freitag, D. (2004). Toward unsupervised whole-corpus tagging. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 357, Morristown, NJ, USA. Association for Computational Linguistics.

Gale, W. A. and Church, K. W. (1994). What's wrong with adding one? In *Corpus-Based Research into Language*, pages 189–198.

Gao, J. and Suzuki, H. (2003). Unsupervised learning of dependency structure for language modeling. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 521–528, Stroudsburg, PA, USA. Association for Computational Linguistics.

Garside, R., Leech, G., and McEnery, A. (1997). *Corpus Annotation*. Longman, London and New York.

Geertzen, J. (2003). String alignment in grammatical induction: What suffix trees can do. ILK Research Group Technical Report Series 03-11, Tilburg University.

Golding, A. and Roth, D. (1999). A Winnow-Based Approach to Context-Sensitive Spelling Correction. *Machine Learning*, 34(1–3):107–130.

Golding, A. R. (1995). A Bayesian hybrid method for context-sensitive spelling correction. In *Proceedings of the 3rd workshop on very large corpora, ACL-95*, pages 39–53.

Good, I. J. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika*, 40:237–264.

Gorla, J., Goyal, A., and Sangal, R. (2007). Two approaches for building an unsupervised dependency parser and their other applications. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 2*, pages 1860–1861, Vancouver, British Columbia, Canada. AAAI Press.

Grishman, R., Macleod, C., and Meyers, A. (1994). Comlex syntax: Building a computational lexicon. In *Proceedings of the 15th International Conference on Computational Linguistics, COLING-94*, pages 268–272, New York. New York University.

Grünwald, P. (1996). A minimum description length approach to grammar inference. In Wermter, S., Riloff, E., and Scheler, G., editors, *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, volume 1040 of *Lecture Notes in Artificial Intelligence*, pages 203–216. Springer-Verlag, Berlin.

Grünwald, P. D. (2007). *The Minimum Description Length Principle*. MIT Press, US.

Guiasu, S. and Shenitzer, A. (1985). The principle of maximum entropy. *The Mathematical Intelligencer*, 7(1).

Gusfield, D. (1997). *Algorithms on Strings, Trees and Sequences*. University of Cambridge, Cambridge.

Hajič, J., Ciaramita, M., Johansson, R., Kawahara, D., Martí, M. A., Màrquez, L., Meyers, A., Nivre, J., Padó, S., Štěpánek, J., Straňák, P., Surdeanu, M., Xue, N., and Zhang, Y. (2009). The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proc. of CoNLL-2009: Shared Task*, pages 1–18, Boulder, Colorado, USA.

Hall, J., Nilsson, J., Nivre, J., Eryigit, G., Megyesi, B., Nilsson, M., and Saers, M. (2007). Single malt or blended? a study in multilingual parser optimization. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 933–939.

Headden III, W. P., Johnson, M., and McClosky, D. (2009). Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 101–109, Boulder, Colorado. Association for Computational Linguistics.

Hirst, G. and Budanitsky, A. (2005). Correcting real-word spelling errors by restoring lexical cohesion. *Natural Language Engineering*, 11(1):87–111.

Huang, J. H. and Powers, D. W. (2001). Large scale experiments on correction of confused words. In *Australasian Computer Science Conference Proceedings*, pages 77–82, Gold Coast, Queensland, Australia. Bond University.

Jelinek, F. (1998). *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, MA.

Jurafsky, D. and Martin, J. H. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, Englewood Cliffs, New Jersey.

Katz, S. M. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-35:400–401.

Kingsbury, P., Palmer, M., and Marcus, M. (2002). Adding semantic annotation to the Penn Treebank. In *Proceedings of the Human Language Technology Conference*, San Diego, CA.

Knuth, D. E. (1973). *The Art of Computer Programming*, volume 3: Sorting and Searching. Addison-Wesley, Reading, MA.

Kukich, K. (1992). Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24(4):377–439.

Kučera, H. and Francis, W. N. (1967). *Computational Analysis of Present-Day American English*. Brown University Press, Providence, RI.

Lapata, M. and Keller, F. (2004). The Web as a Baseline: Evaluating the Performance of Unsupervised Web-based Models for a Range of NLP Tasks. In Dumais, S., Marcu, D., and Roukos, S., editors, *HLT-NAACL 2004: Main Proceedings*, pages 121–128, Boston, MA. Association for Computational Linguistics.

Lee, J. and Seneff, S. (2006). Automatic Grammar Correction for Second-Language Learners. In *Ninth International Conference on Spoken Language Processing*, pages 1978–1981. ISCA.

Lee, J. and Seneff, S. (2008). Correcting misuse of verb forms. In *Proceedings of ACL-08: HLT*, pages 174–182, Columbus, Ohio. Association for Computational Linguistics.

Leech, G., Garside, R., and Bryant, M. (1994). Claws4: The tagging of the british national corpus.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Sovjet Physics Doklady*, 10:707–710.

Lewis, D., Yang, Y., Rose, T., and Li, F. (2004). RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397.

Magerman, D. and Marcus, M. (1990). Parsing a natural language using mutual information statistics. In *Proceedings of 8th. conference on AI (AAAI-90)*, volume 2, pages 984–989.

Malouf, R. (2000). The order of prenominal adjectives in natural language generation. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 85–92, New Brunswick, NJ. ACL.

Manber, U. and Myers, G. (1990). Suffix arrays: a new method for on-line string searches. In *SODA '90: Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, pages 319–327, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Mangu, L. and Brill, E. (1997). Automatic rule acquisition for spelling correction. In *Proceedings of the International Conference on Machine Learning*, pages 187–194.

Manning, C. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA.

Manzini, G. and Ferragina, P. (2004). Engineering a lightweight suffix array construction algorithm. *Algorithmica*, 40:33–50.

Marcus, M., Santorini, S., and Marcinkiewicz, M. (1993). Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Markov, A. A. (2006). An example of statistical investigation of the text Eugene Onegin concerning the connection of samples in chains. *Science in Context*, 19(4):591–600.

Màrquez, L. and Padró, L. (1997). A flexible pos tagger using an automatically acquired language model. In *Proceedings of EACL/ACL 1997*, pages 238–245, Madrid, Spain.

Mays, E., Damerau, F. J., and Mercer, R. L. (1991). Context based spelling correction. *Information Processing and Management*, 27(5):517–522.

McDonald, R. and Nivre, J. (2007). Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages pp. 122–131. Association for Computational Linguistics.

McDonald, R. and Pereira, F. (2006). Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 81–88.

McDonald, R., Pereira, F., Ribarov, K., and Hajič, J. (2005). Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530.

McNemar, Q. (1947). Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages. *Psychometrika*, 12(2):153–157.

Miller, G., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K. (1990). Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–312.

Mitchell, M. (2009). Class-based ordering of prenominal modifiers. In *Proceedings of the 12th European Workshop on Natural Language Generation*, ENLG '09, pages 50–57, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mitton, R. (1996). *English Spelling and the Computer*. Longman, Harlow, Essex, UK.

Morante, R., Van Asch, V., and Van den Bosch, A. (2009a). Dependency parsing and semantic role labeling as a single task. In *Proceedings of the 7th International Conference on Recent Advances in Natural Language Processing (RANLP-2009),*, pages 275–280, Borovets, Bulgaria.

Morante, R., Van Asch, V., and Van den Bosch, A. (2009b). Joint memory-based learning of syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL): Shared Task,*, pages 25–30, Boulder, CO, USA.

Morrison, D. R. (1968). Patricia—practical algorithm to retrieve information coded in alphanumeric. *J. ACM*, 15(4):514–534.

Nivre, J. (2003). An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160.

Nivre, J., Hall, J., Kübler, S., McDonald, R., Nilsson, J., Riedel, S., and Yuret, D. (2007a). The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, Prague, Czech Republic. Association for Computational Linguistics.

Nivre, J., Hall, J., and Nilsson, J. (2004). Memory-based dependency parsing. In Ng, H. T. and Riloff, E., editors, *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL 2004)*, pages 49–57, Boston, MA.

Nivre, J., Hall, J., and Nilsson, J. (2006). Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC-2006*, pages 2216–2219.

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S., and Marsi, E. (2007b). MaltParser: a language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Nivre, J. and McDonald, R. (2008). Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, pages 950–958, Columbus, Ohio. Association for Computational Linguistics.

Nøklestad, A. (2009). *A Machine Learning Approach to Anaphora Resolution Including Named Entity Recognition, PP Attachment Disambiguation, and Animacy Detection*. PhD thesis, University of Oslo.

Oostdijk, N., Reynaert, M., Monachesi, P., Van Noord, G., Ordelman, R., Schuurman, I., and Vandeghinste, V. (2008). From D-Coi to SoNaR: A reference corpus for Dutch. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco.

R Development Core Team (2010). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Ratnaparkhi, A. (1996). A maximum entropy part-of-speech tagger. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, May 17-18, 1996, University of Pennsylvania*.

Reynaert, M. (2005). *Text-induced spelling correction*. PhD thesis, Tilburg University.

Reynaert, M. (2008). All, and only, the errors: More complete and consistent spelling and OCR-error correction evaluation. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco.

Sandra, D., Daems, F., and Frisson, S. (2001). Zo helder en toch zoveel fouten! wat leren we uit psycholinguïstisch onderzoek naar werkwoordfouten bij ervaren spellers? *Tijdschrift van de Vereniging voor het Onderwijs in het Nederlands*, 30(3):3–20.

Schaback, J. and Li, F. (2007). Multi-level feature extraction for spelling correction. In *IJCAI-2007 Workshop on Analytics for Noisy Unstructured Text Data*, pages 79–86, Hyderabad, India.

Schütze, H. (1993). Part-of-speech induction from scratch. In *Proceedings of the Annual Meeting of the ACL*, pages 251–258.

Seginer, Y. (2007). Fast unsupervised incremental parsing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 384–391, Prague, Czech Republic. Association for Computational Linguistics.

Shaw, J. and Hatzivassiloglou, V. (1999). Ordering among premodifiers. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 135–143, College Park, Maryland, USA. Association for Computational Linguistics.

Spitkovsky, V. I., Alshawi, H., and Jurafsky, D. (2009). Baby Steps: How "Less is More" in unsupervised dependency parsing. In *NIPS: Grammar Induction, Representation of Language and Language Learning*, pages 1–10.

Steetskamp, R. (1995). An implementation of a probabilistic tagger. Master's thesis, TOSCA Research Group, University of Nijmegen, Nijmegen, The Netherlands.

Stehouwer, H. and Van den Bosch, A. (2009). Putting the t where it belongs: Solving a confusion problem in Dutch. In Verberne, S., van Halteren, H., and Coppen, P.-A., editors, *Computational Linguistics in the Netherlands 2007: Selected Papers from the 18th CLIN Meeting*, pages 21–36, Nijmegen, The Netherlands.

Stehouwer, H. and Van Zaanen, M. (2009a). Language models for contextual error detection and correction. In *Proceedings of the EACL 2009 Workshop on Computational Linguistic Aspects of Grammatical Inference*, pages 41–48, Athens, Greece.

Stehouwer, H. and Van Zaanen, M. (2009b). Token merging in language model-based confusible disambiguation. In *Proceedings of the 21st Benelux Conference on Artificial Intelligence (BNAIC-2009)*, pages 241–248, Eindhoven, The Netherlands.

Stehouwer, H. and Van Zaanen, M. (2010a). Enhanced suffix arrays as language models: Virtual k-testable languages. In Sempere, J. and García, P., editors, *Grammatical Inference: Theoretical Results and Applications*, volume 6339 of *Lecture Notes in Computer Science*, pages 305–308. Springer Berlin / Heidelberg.

Stehouwer, H. and Van Zaanen, M. (2010b). Finding patterns in strings using suffixarrays. In *Proceedings of Computational Linguistics—Applications, 2010*, pages 151–158. International Multiconference on Computer Science and Information Technology.

Stehouwer, H. and Van Zaanen, M. (2010c). Using suffix arrays as language models: Scaling the n-gram. In *Proceedings of the 22nd Benelux Conference on Artificial Intelligence (BNAIC)*.

Steinbach, M., Karypis, G., and Kumar, V. (2000). A comparison of document clustering techniques. In *KDD Workshop on Text Mining*.

Stolcke, A. (2002). SRILM – An extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, pages 901–904, Denver, Colorado.

Sun, G., Liu, X., Cong, G., Zhou, M., Xiong, Z., Lee, J., and Lin, C. (2007). Detecting erroneous sentences using automatically mined sequential patterns. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 81–88, Prague, Czech Republic. Association for Computational Linguistics.

Surdeanu, M., Johansson, R., Meyers, A., Màrquez, L., and Nivre, J. (2008). The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proc. of CoNLL-2008*.

Ukkonen, E. (1995). On-line construction of suffix trees. *Algorithmica*, 14(3):249–260.

Van Delden, S., Bracewell, D. B., and Gomez, F. (2004). Supervised and unsupervised automatic spelling correction algorithms. In Zhang, D., Grégoire, É., and DeGroot, D., editors, *Proceedings of the 2004 IEEE International Conference on Information Reuse and Integration*, pages 530–535, Las Vegas, NV.

Van den Bosch, A. (2006a). All-word prediction as the ultimate confusible disambiguation. In *Proceedings of the HLT-NAACL Workshop on Computationally hard problems and joint inference in speech and language processing*, New York, NY.

Van den Bosch, A. (2006b). Scalable classification-based word prediction and confusible correction. *Traitement Automatique des Langues*, 46(2):39–63.

Van den Bosch, A. (2009). Machine learning. In Lüdeling, A. and Kytö, M., editors, *Corpus Linguistics: An International Handbook*, volume 2, pages 855–872. Walter de Gruyter, Berlin.

Van den Bosch, A. and Buchholz, S. (2002). Shallow parsing on the basis of words only: A case study. In *Proceedings of the 40th Meeting of the Association for Computational Linguistics*, pages 433–440.

Van den Bosch, A., Busser, G., Canisius, S., and Daelemans, W. (2007). An efficient memory-based morpho-syntactic tagger and parser for Dutch. In Dirix, P., Schuurman, I., Vandeghinste, V., and Van Eynde, F., editors, *Computational Linguistics in the Netherlands: Selected Papers from the Seventeenth CLIN Meeting*, pages 99–114, Leuven, Belgium.

Van der Beek, L., Bouma, G., Malouf, R., and Van Noord, G. (2001). The Alpino Dependency Treebank. In *Selected Papers from the Twelfth Computational Linguistics in the Netherlands Meeting, CLIN-2001*, Amsterdam, The Netherlands. Rodopi.

Van Eynde, F. (2004). Part of speech tagging en lemmatisering van het Corpus Gesproken Nederlands. Technical report, Centrum voor Computerlinguïstiek, K.U. Leuven.

Van Eynde, F., Zavrel, J., and Daelemans, W. (2000). Part of speech tagging and lemmatisation for the Spoken Dutch Corpus. In *In Proceedings of LREC'2000*, pages 1427–1433.

Van Rijsbergen, C. (1979). *Information Retrieval*. Buttersworth, London.

Van Zaanen, M. (2000). ABL: Alignment-Based Learning. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING); Saarbrücken, Germany*, pages 961–967. Association for Computational Linguistics.

Van Zaanen, M. (2010). Suffix tree package. http://ilk.uvt.nl/ menno/research/-software/suffixtree.

Vandekerckhove, B., Sandra, D., and Daelemans, W. (2011). Selective impairment of adjective order constraints as overeager abstraction: An expansion on Kemmerer et al. (2009). Manuscript submitted for publication.

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269.

Warnock, T. and Wendroff, B. (1988). Search tables in computer chess. In *ICCA Journal*, volume 11–1, pages 10–13.

Weiss, S. and Kulikowski, C. (1991). *Computer systems that learn*. San Mateo, CA: Morgan Kaufmann.

Wilcox-O'Hearn, L. A., Hirst, G., and Budanitsky, A. (2008). Real-Word spelling correction with trigrams: A reconsideration of the Mays, Damerau, and Mercer model. In Gelbukh, A., editor, *Proceedings of the Computational Linguistics and Intelligent Text Processing 9th International Conference, CICLing 2008*, volume LNCS 4919, pages 605–616, Berlin, Germany. Springer Verlag.

Wu, D., Sui, Z., and Zhao, J. (1999). An information-based method for selecting feature types for word prediction. In *Proceedings of the Sixth European Conference on Speech Communication and Technology, EUROSPEECH'99*, Budapest.

Wu, M.-W. and Su, K.-Y. (1993). Corpus-based compound extraction with mutual information and relative frequency count. In *Proceedings of ROCLING VI*, pages 207–216.

Yamamoto, M. and Church, K. (2001). Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus. *Computational Linguistics*, 27:28–37.

Yarowsky, D. (1994). Decision lists for lexical ambiguity resolution: application to accent restoration in Spanish and French. In *Proceedings of the Annual Meeting of the ACL*, pages 88–95.

Younger, D. (1967). Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.

Zavrel, J. and Daelemans, W. (1999). Recent advances in memory-based part-of-speech tagging. In *VI Simposio Internacional de Comunicacion Social*, pages 590–597.

Zobrist, A. L. (1970). A new hashing method with application for game playing. In *ICCA Journal (republished)*, volume 13–2, pages 69–73.

Summary

Over the last decades, language modelling has played an important part in natural language processing. However, the flexibility of the language models has not increased over time, and problems with data sparseness persist. Our research is motivated by our desire to provide a language model paradigm that is flexible and less effected by sparseness.

Derived from the motivation for the research, we phrase the following two problem statements: (PS1) *Is it helpful to create a statistical language model that is flexible, i.e., not fixed in advance, with regards to the n-gram size, for adequately handling the problem of sparseness?*, and (PS2) *Do linguistically motivated annotations and their automatically generated counterparts provide information that can be successfully used as a back-off step to handle sparseness? Does alleviating sparseness in this way increase performance on alternative sequence selection tasks?* To address the problem statements, we investigate several language model environments on three alternative sequence selection problems.

As language model environments we investigate (1) a flexible language model (in terms of the size of the n-grams used) without annotation, (2) a flexible language model enriched with either the human-designed or machine-derived part-of-speech tags, and (3) a flexible language model enriched with human-designed or machine-derived dependency information. These three language model environments are each used on a set of three alternative sequence selection problems. To wit, (1) confusibles, (2) verb and noun agreement, (3) prenominal adjective reordering.

The proposed investigations can be performed by finding answers to four research questions: (RQ1) *Is there a need to predetermine or limit the size of the n-grams used in language models? Is there an inherent advantage or disadvantage to using a fixed-size n?*, (RQ2) *If the size of the n-grams is not fixed in advance, how can we still generate comparable distributions when we select among alternative sequences?*, (RQ3) *Is there a benefit to including annotations in the language model, measured as a better performance on alternative*

sequence selection tasks?, and (RQ4) *Is there a difference in performance on alternative sequence selection tasks when using human-designed annotations compared to machine-generated annotations?*.

We provide background on the three selection problems in Chapter 2. This is followed by a detailed overview of our experimental setup in Chapter 3. This experimental setup is used throughout this thesis to examine the effects of the different language model environments on the three alternative selection problems. These results form the basis for our answers to the research questions and problem statements.

We dedicate one chapter to each of the three language model environments used, starting with Chapter 4 for the language model environment without any added annotations (RQ1 and RQ2). In the chapter we provide literary background on language models, present some preliminary work that shaped the design of our language model system, and we present the results of the language model environment without any added annotation. These results lead us to conclude that by synchronising the distributions used on the different alternatives we can increase the flexibility of the language model system as well as outperform a range of existing language models on the three alternative selection tasks..

In Chapter 5 we discuss background and results of the language model environment that can be characterised by the addition of part-of-speech annotation (RQ3 and RQ4). We examine two annotations, (1) the human-designed part-of-speech annotation, and (2) a machine-derived part-of-speech annotation. The results lead us to conclude that (1) the machine-generated part-of-speech tagset does not add any new information to the model, (2) the human-designed part-of-speech tagset does add new information to the model, and (3) as the size of the training material increases the addition of the part-of-speech tags contributes less and less to the performance of the system. As we observed the point where the addition of part-of-speech tags no longer adds to the system occurs around one million sentences of training material for all three alternative sequence selection tasks.

In Chapter 6 we discuss background and results of the language model environment that can be characterised by the addition of dependency information (RQ3 and RQ4). We examine two annotations, (1) added human-designed dependency information using the Maltparser, and (2) added machine-derived dependency information using the common-cover-link (CCL) parser. The results lead us to conclude that, (1) dependency annotations, when employed as in this thesis, hamper the performance of the model, and (2) when using a merged token combining the annotation and the word token in the focus position the model performs considerable worse, due to sparseness.

In Chapter 7 we conclude the thesis by answering the four research questions and the two problem statements. Given that the benefits of adding annotation to language models lessen when more data becomes available, we conclude that (1) we have created a flexible language model that automatically determines the optimal n-gram size, and (2) while integrating added annotations in a flexible and automatic manner is possible, the benefits of annotations disappear with enough training material. Also, we conclude about the use of machine-derived annotations that, in combination with synchronous back-off, they add no novel information to the language model as synchronous back-off implicitly derived similar information. In addition to the conclusions, Chapter 7 presents possible avenues for future research.

Samenvatting

In de laatste decennia hebben statistische taalmodellen een grote rol gespeeld binnen de computationele taalkunde. De flexibiliteit van deze taalmodellen is in de loop van de tijd echter niet toegenomen. Ook zijn er nog immer problemen met de schaarste van de data. Ons onderzoek is gemotiveerd door het verlangen om statische taalmodellen flexibeler te maken en minder last te hebben van de schaarste aan data.

Op basis van deze motivatie en met kennis van de relevante literatuur hebben we twee probleemstellingen gedefinieerd: (PS1) *In hoeverre is het nuttig een flexibeler statistisch taalmodel te maken, dwz., een taalmodel waarbij de grootte van de n-grammen niet van te voren is vastgelegd, voor het adequaat omgaan met het schaarste-probleem?* en (PS2) *Helpen taalkundig gemotiveerde annotaties en hun machinaal-gegenereerde equivalenten als back-off stap tegen het schaarste-probleem van de data? Geeft het aanpakken van schaarste op deze manier een voordeel bij alternatieve sequentieselectie-taken?* Om deze vragen te beantwoorden onderzoeken we drie verschillende statistische taalmodellen over drie alternatieve sequentieselectie-problemen. Het gaat om de volgende taalmodellen: (1) een flexibel taalmodel zonder toegevoegde annotaties, (2) een flexibel taalmodel dat verrijkt is met òf menselijk-gedefinieerde òf machinaal-gegenereerde woordsoort-annotatie, en (3) een flexibel taalmodel dat verrijkt is met òf menselijk-gedefinieerde òf machinaal-gegenereerde dependency-annotatie. Deze drie taalmodellen worden toegepast op drie alternatieve sequentieselectie-problemen, namelijk (1) confusibles, (2) werkwoord en zelfstandig naamwoord congruentie, en (3) ordening van prenominale adjectieven.

In Hoofdstuk 2 geven we achtergrondinformatie over de drie alternatieve sequentieselectie-problemen. Dit wordt direct gevolgd door een overzicht van onze experimentele opzet in Hoofdstuk 3. Deze opzet is voor het gehele proefschrift hetzelfde om de effecten van het gebruik van de verschillende taalmodellen zo precies mogelijk te bestuderen.

Ieder taalmodel wordt in één hoofdstuk beschreven te samen met de resultaten. We beginnen daarmee in Hoofdstuk 4 voor het flexibele taalmodel zonder annotaties. In dit hoofdstuk geven we enige theoretische achtergrond, beschrijven we eerdere experimenten die het ontwerp van de taalmodellen hebben beïnvloed, en presenteren we de resultaten van het taalmodel zonder annotaties. De resultaten leiden ons tot de conclusie dat we door het synchroniseren van de distributies die de taalmodellen gebruiken flexibel kunnen zijn in de grootte van de gebruikte n-grammen voor alternatieve sequentieselectie-problemen.

Vervolgens beschrijven we in Hoofdstuk 5 eerst enige achtergrond en geven daarna de resultaten die gerelateerd zijn aan het taalmodel met toegevoegde woordsoort-annotaties. Specifiek kijken we naar het effect van twee verschillende annotaties, (1) een menselijk-gedefinieerde woordsoort-annotatie, en (2) een machine-gegenereerde woordsoort-annotatie. Op basis van de observaties gemaakt in dit hoofdstuk concluderen wij het volgende: (1) de machine-gegenereerde annotatie voegt geen nieuwe, bruikbare informatie toe aan het model, (2) de menselijk-gedefinieerde annotatie voegt wel nieuwe, bruikbare informatie toe aan het model, en (3) als de hoeveelheid van het trainingmateriaal toeneemt, neemt de toegevoegde waarde van de woordsoort-annotatie af.

In Hoofdstuk 6 geven we opnieuw enige achtergrond en de resultaten, dit keer voor het taalmodel met toegevoegde dependency-annotatie. We bestuderen wederom twee annotaties: (1) een menselijk-gedefinieerde dependency-annotatie die wordt toegevoegd door de Maltparser en (2) een machine-gegenereerde dependency-annotatie die wordt toegevoegd door de common-cover-link (CCL) parser. Op basis van de observaties gemaakt in dit hoofdstuk concluderen wij het volgende: (1) dependency-annotatie, zoals hier gebruikt, heeft geen positieve invloed op de prestaties van het model en (2) als we zowel het tekst-token als het annotatie-token gebruiken op de focus-positie is de prestatie van het model aanmerkelijk slechter.

In Hoofdstuk 7 formuleren we de conclusie van dit proefschrift in de vorm van antwoorden op de onderzoeksvragen, waarmee we eveneens de probleemstellingen beantwoorden. Uit de observaties dat de voordelen van het toevoegen van annotaties aan taalmodellen afnemen als er meer training materiaal beschikbaar komt, mogen we concluderen dat (1) we een flexibel taalmodel hebben gemaakt dat op een automatische manier de optimale grootte van het n-gram bepaalt en (2) hoewel het mogelijk is om automatisch en flexibel annotaties te integreren de voordelen daarvan gering zijn als er voldoende trainingmateriaal aanwezig is. Ook merken we op dat het gebruik van machine-gegenereerde annotaties, in combinatie met synchronous back-off, geen nieuwe informatie toevoegt aan het model, aangezien synchronous back-off zelf al impliciet gelijksoortige informatie afleidt. Tevens wordt in Hoofdstuk 7 een richting aangegeven voor toekomstig onderzoek.

Curriculum Vitae

Johan Herman Stehouwer was born in Delfzijl, on the 16th of June 1983. From 1994 to 2001 he attended the Ommelander College (Atheneum Nature and Technology). From 2001 to 2006 he studied Computer Science at Twente University, where he majored on Machine Learning on Natural Language (on a comparison of a dozen models for selecting key phrases in running text).

Directly following his studies at Twente University he was given the opportunity to pursue a Ph.D. at Tilburg University, starting in September 2006. There he participated in the NWO Vici project Implicit Linguistics. Here he explored several subject areas such as Machine Translation and Spelling Correction. In 2008 he took up the research resulting in this thesis, on Language Models for alternative sequence selection.

Since February 2011 he works as a scientific software developer for the Max Planck Institute for Psycholinguistics.

Publications

The scientific work performed during the author's Ph.D. research resulted in the following publications.

Conference and workshop proceedings

1. Stehouwer, H. and Van den Bosch, A. (2009). Putting the t where it belongs: Solving a confusion problem in Dutch. In Verberne, S., van Halteren, H., and Coppen, P.-A., editors, *Computational Linguistics in the Netherlands 2007: Selected Papers from the 18th CLIN Meeting*, pages 21–36, Nijmegen, The Netherlands

2. Stehouwer, H. and Van Zaanen, M. (2009a). Language models for contextual error detection and correction. In *Proceedings of the EACL 2009 Workshop on Computational Linguistic Aspects of Grammatical Inference*, pages 41–48, Athens, Greece

3. Stehouwer, H. and Van Zaanen, M. (2009b). Token merging in language model-based confusible disambiguation. In *Proceedings of the 21st Benelux Conference on Artificial Intelligence (BNAIC-2009)*, pages 241–248, Eindhoven, The Netherlands

4. Stehouwer, H. and Van Zaanen, M. (2010a). Enhanced suffix arrays as language models: Virtual k-testable languages. In Sempere, J. and García, P., editors, *Grammatical Inference: Theoretical Results and Applications*, volume 6339 of *Lecture Notes in Computer Science*, pages 305–308. Springer Berlin / Heidelberg

5. Stehouwer, H. and Van Zaanen, M. (2010b). Finding patterns in strings using suffixarrays. In *Proceedings of Computational Linguistics—Applications, 2010*, pages 151–158. International Multiconference on Computer Science and Information Technology

6. Stehouwer, H. and Van Zaanen, M. (2010c). Using suffix arrays as language models: Scaling the n-gram. In *Proceedings of the 22nd Benelux Conference on Artificial Intelligence (BNAIC)*

SIKS Dissertation Series

1998

1 Johan van den Akker (CWI) *DEGAS - An Active, Temporal Database of Autonomous Objects*

2 Floris Wiesman (UM) *Information Retrieval by Graphically Browsing Meta-Information*

3 Ans Steuten (TUD) *A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective*

4 Dennis Breuker (UM) *Memory versus Search in Games*

5 Eduard W. Oskamp (RUL) *Computerondersteuning bij Straftoemeting*

1999

1 Mark Sloof (VU) *Physiology of Quality Change Modelling; Automated Modelling of Quality Change of Agricultural Products*

2 Rob Potharst (EUR) *Classification using Decision Trees and Neural Nets*

3 Don Beal (UM) *The Nature of Minimax Search*

4 Jacques Penders (UM) *The Practical Art of Moving Physical Objects*

5 Aldo de Moor (KUB) *Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems*

6 Niek J.E. Wijngaards (VU) *Re-Design of Compositional Systems*

7 David Spelt (UT) *Verification Support for Object Database Design*

8 Jacques H.J. Lenting (UM) *Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation*

2000

1 Frank Niessink (VU) *Perspectives on Improving Software Maintenance*

2 Koen Holtman (TU/e) *Prototyping of CMS Storage Management*

3 Carolien M.T. Metselaar (UvA) *Sociaal-organisatorische Gevolgen van Kennistechnologie; een Procesbenadering en Actorperspectief*

4 Geert de Haan (VU) *ETAG, A Formal Model of Competence Knowledge for User Interface Design*

5 Ruud van der Pol (UM) *Knowledge-Based Query Formulation in Information Retrieval*

6 Rogier van Eijk (UU) *Programming Languages for Agent Communication*

7 Niels Peek (UU) *Decision-Theoretic Planning of Clinical Patient Management*

Abbreviations. SIKS – Dutch Research School for Information and Knowledge Systems; CWI – Centrum voor Wiskunde en Informatica, Amsterdam; DROP – Delft Research institute for Operations Programming; EUR – Erasmus Universiteit, Rotterdam; KUB – Katholieke Universiteit Brabant, Tilburg; KUN – Katholieke Universiteit Nijmegen; OU – Open Universiteit Nederland; RUG – Rijksuniversiteit Groningen; RUL – Rijksuniversiteit Leiden; RUN – Radboud Universiteit Nijmegen; TUD – Technische Universiteit Delft; TU/e – Technische Universiteit Eindhoven; UL – Universiteit Leiden; UM – Universiteit Maastricht; UT – Universiteit Twente; UU – Universiteit Utrecht; UvA – Universiteit van Amsterdam; UvT – Universiteit van Tilburg; VU – Vrije Universiteit, Amsterdam.

8 Veerle Coupé (EUR) *Sensitivity Analyis of Decision-Theoretic Networks*
9 Florian Waas (CWI) *Principles of Probabilistic Query Optimization*
10 Niels Nes (CWI) *Image Database Management System Design Considerations, Algorithms and Architecture*
11 Jonas Karlsson (CWI) *Scalable Distributed Data Structures for Database Management*

2001

1 Silja Renooij (UU) *Qualitative Approaches to Quantifying Probabilistic Networks*
2 Koen Hindriks (UU) *Agent Programming Languages: Programming with Mental Models*
3 Maarten van Someren (UvA) *Learning as Problem Solving*
4 Evgueni Smirnov (UM) *Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets*
5 Jacco van Ossenbruggen (VU) *Processing Structured Hypermedia: A Matter of Style*
6 Martijn van Welie (VU) *Task-Based User Interface Design*
7 Bastiaan Schonhage (VU) *Diva: Architectural Perspectives on Information Visualization*
8 Pascal van Eck (VU) *A Compositional Semantic Structure for Multi-Agent Systems Dynamics*
9 Pieter Jan 't Hoen (RUL) *Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes*
10 Maarten Sierhuis (UvA) *Modeling and Simulating Work Practice BRAHMS: a Multiagent Modeling and Simulation Language for Work Practice Analysis and Design*
11 Tom M. van Engers (VU) *Knowledge Management: The Role of Mental Models in Business Systems Design*

2002

1 Nico Lassing (VU) *Architecture-Level Modifiability Analysis*
2 Roelof van Zwol (UT) *Modelling and Searching Web-based Document Collections*
3 Henk Ernst Blok (UT) *Database Optimization Aspects for Information Retrieval*
4 Juan Roberto Castelo Valdueza (UU) *The Discrete Acyclic Digraph Markov Model in Data Mining*
5 Radu Serban (VU) *The Private Cyberspace Modeling Electronic Environments Inhabited by Privacy-Concerned Agents*
6 Laurens Mommers (UL) *Applied Legal Epistemology; Building a Knowledge-based Ontology of the Legal Domain*
7 Peter Boncz (CWI) *Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications*
8 Jaap Gordijn (VU) *Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas*
9 Willem-Jan van den Heuvel (KUB) *Integrating Modern Business Applications with Objectified Legacy Systems*
10 Brian Sheppard (UM) *Towards Perfect Play of Scrabble*
11 Wouter C.A. Wijngaards (VU) *Agent Based Modelling of Dynamics: Biological and Organisational Applications*
12 Albrecht Schmidt (UvA) *Processing XML in Database Systems*
13 Hongjing Wu (TU/e) *A Reference Architecture for Adaptive Hypermedia Applications*
14 Wieke de Vries (UU) *Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems*
15 Rik Eshuis (UT) *Semantics and Verification of UML Activity Diagrams for Workflow Modelling*
16 Pieter van Langen (VU) *The Anatomy of Design: Foundations, Models and Applications*
17 Stefan Manegold (UvA) *Understanding, Modeling, and Improving Main-Memory Database Performance*

2003

1 Heiner Stuckenschmidt (VU) *Ontology-Based Information Sharing in Weakly Structured Environments*
2 Jan Broersen (VU) *Modal Action Logics for Reasoning About Reactive Systems*
3 Martijn Schuemie (TUD) *Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy*
4 Milan Petkovic (UT) *Content-Based Video Retrieval Supported by Database Technology*

5 Jos Lehmann (UvA) *Causation in Artificial Intelligence and Law – A Modelling Approach*
6 Boris van Schooten (UT) *Development and Specification of Virtual Environments*
7 Machiel Jansen (UvA) *Formal Explorations of Knowledge Intensive Tasks*
8 Yong-Ping Ran (UM) *Repair-Based Scheduling*
9 Rens Kortmann (UM) *The Resolution of Visually Guided Behaviour*
10 Andreas Lincke (UT) *Electronic Business Negotiation: Some Experimental Studies on the Interaction between Medium, Innovation Context and Cult*
11 Simon Keizer (UT) *Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks*
12 Roeland Ordelman (UT) *Dutch Speech Recognition in Multimedia Information Retrieval*
13 Jeroen Donkers (UM) *Nosce Hostem – Searching with Opponent Models*
14 Stijn Hoppenbrouwers (KUN) *Freezing Language: Conceptualisation Processes across ICT-Supported Organisations*
15 Mathijs de Weerdt (TUD) *Plan Merging in Multi-Agent Systems*
16 Menzo Windhouwer (CWI) *Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouse*
17 David Jansen (UT) *Extensions of Statecharts with Probability, Time, and Stochastic Timing*
18 Levente Kocsis (UM) *Learning Search Decisions*

2004

1 Virginia Dignum (UU) *A Model for Organizational Interaction: Based on Agents, Founded in Logic*
2 Lai Xu (UvT) *Monitoring Multi-party Contracts for E-business*
3 Perry Groot (VU) *A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving*
4 Chris van Aart (UvA) *Organizational Principles for Multi-Agent Architectures*
5 Viara Popova (EUR) *Knowledge Discovery and Monotonicity*
6 Bart-Jan Hommes (TUD) *The Evaluation of Business Process Modeling Techniques*
7 Elise Boltjes (UM) *Voorbeeld$_{IG}$ Onderwijs; Voorbeeldgestuurd Onderwijs, een Opstap naar Abstract Denken, vooral voor Meisjes*
8 Joop Verbeek (UM) *Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale Politiële Gegevensuitwisseling en Digitale Expertise*
9 Martin Caminada (VU) *For the Sake of the Argument; Explorations into Argument-based Reasoning*
10 Suzanne Kabel (UvA) *Knowledge-rich Indexing of Learning-objects*
11 Michel Klein (VU) *Change Management for Distributed Ontologies*
12 The Duy Bui (UT) *Creating Emotions and Facial Expressions for Embodied Agents*
13 Wojciech Jamroga (UT) *Using Multiple Models of Reality: On Agents who Know how to Play*
14 Paul Harrenstein (UU) *Logic in Conflict. Logical Explorations in Strategic Equilibrium*
15 Arno Knobbe (UU) *Multi-Relational Data Mining*
16 Federico Divina (VU) *Hybrid Genetic Relational Search for Inductive Learning*
17 Mark Winands (UM) *Informed Search in Complex Games*
18 Vania Bessa Machado (UvA) *Supporting the Construction of Qualitative Knowledge Models*
19 Thijs Westerveld (UT) *Using generative probabilistic models for multimedia retrieval*
20 Madelon Evers (Nyenrode) *Learning from Design: facilitating multidisciplinary design teams*

2005

1 Floor Verdenius (UvA) *Methodological Aspects of Designing Induction-Based Applications*
2 Erik van der Werf (UM) *AI techniques for the game of Go*
3 Franc Grootjen (RUN) *A Pragmatic Approach to the Conceptualisation of Language*
4 Nirvana Meratnia (UT) *Towards Database Support for Moving Object data*
5 Gabriel Infante-Lopez (UvA) *Two-Level Probabilistic Grammars for Natural Language Parsing*
6 Pieter Spronck (UM) *Adaptive Game AI*

7 Flavius Frasincar (TU/e) *Hypermedia Presentation Generation for Semantic Web Information Systems*
8 Richard Vdovjak (TU/e) *A Model-driven Approach for Building Distributed Ontology-based Web Applications*
9 Jeen Broekstra (VU) *Storage, Querying and Inferencing for Semantic Web Languages*
10 Anders Bouwer (UvA) *Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments*
11 Elth Ogston (VU) *Agent Based Matchmaking and Clustering - A Decentralized Approach to Search*
12 Csaba Boer (EUR) *Distributed Simulation in Industry*
13 Fred Hamburg (UL) *Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen*
14 Borys Omelayenko (VU) *Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics*
15 Tibor Bosse (VU) *Analysis of the Dynamics of Cognitive Processes*
16 Joris Graaumans (UU) *Usability of XML Query Languages*
17 Boris Shishkov (TUD) *Software Specification Based on Re-usable Business Components*
18 Danielle Sent (UU) *Test-selection strategies for probabilistic networks*
19 Michel van Dartel (UM) *Situated Representation*
20 Cristina Coteanu (UL) *Cyber Consumer Law, State of the Art and Perspectives*
21 Wijnand Derks (UT) *Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics*

2006

1 Samuil Angelov (TU/e) *Foundations of B2B Electronic Contracting*
2 Cristina Chisalita (VU) *Contextual issues in the design and use of information technology in organizations*
3 Noor Christoph (UvA) *The role of metacognitive skills in learning to solve problems*
4 Marta Sabou (VU) *Building Web Service Ontologies*
5 Cees Pierik (UU) *Validation Techniques for Object-Oriented Proof Outlines*
6 Ziv Baida (VU) *Software-aided Service Bundling - Intelligent Methods & Tools for Graphical Service Modeling*
7 Marko Smiljanic (UT) *XML schema matching – balancing efficiency and effectiveness by means of clustering*
8 Eelco Herder (UT) *Forward, Back and Home Again - Analyzing User Behavior on the Web*
9 Mohamed Wahdan (UM) *Automatic Formulation of the Auditor's Opinion*
10 Ronny Siebes (VU) *Semantic Routing in Peer-to-Peer Systems*
11 Joeri van Ruth (UT) *Flattening Queries over Nested Data Types*
12 Bert Bongers (VU) *Interactivation - Towards an e-cology of people, our technological environment, and the arts*
13 Henk-Jan Lebbink (UU) *Dialogue and Decision Games for Information Exchanging Agents*
14 Johan Hoorn (VU) *Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change*
15 Rainer Malik (UU) *CONAN: Text Mining in the Biomedical Domain*
16 Carsten Riggelsen (UU) *Approximation Methods for Efficient Learning of Bayesian Networks*
17 Stacey Nagata (UU) *User Assistance for Multitasking with Interruptions on a Mobile Device*
18 Valentin Zhizhkun (UvA) *Graph transformation for Natural Language Processing*
19 Birna van Riemsdijk (UU) *Cognitive Agent Programming: A Semantic Approach*
20 Marina Velikova (UvT) *Monotone models for prediction in data mining*
21 Bas van Gils (RUN) *Aptness on the Web*
22 Paul de Vrieze (RUN) *Fundaments of Adaptive Personalisation*
23 Ion Juvina (UU) *Development of Cognitive Model for Navigating on the Web*
24 Laura Hollink (VU) *Semantic Annotation for Retrieval of Visual Resources*
25 Madalina Drugan (UU) *Conditional log-likelihood MDL and Evolutionary MCMC*
26 Vojkan Mihajlovic (UT) *Score Region Algebra: A Flexible Framework for Structured Information Retrieval*
27 Stefano Bocconi (CWI) *Vox Populi: generating video documentaries from semantically annotated media repositories*

28 Borkur Sigurbjornsson (UvA) *Focused Information Access using XML Element Retrieval*

2007

1 Kees Leune (UvT) *Access Control and Service-Oriented Architectures*
2 Wouter Teepe (RUG) *Reconciling Information Exchange and Confidentiality: A Formal Approach*
3 Peter Mika (VU) *Social Networks and the Semantic Web*
4 Jurriaan van Diggelen (UU) *Achieving Semantic Interoperability in Multi-agent Systems: a dialogue-based approach*
5 Bart Schermer (UL) *Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance*
6 Gilad Mishne (UvA) *Applied Text Analytics for Blogs*
7 Natasa Jovanovic' (UT) *To Whom It May Concern - Addressee Identification in Face-to-Face Meetings*
8 Mark Hoogendoorn (VU) *Modeling of Change in Multi-Agent Organizations*
9 David Mobach (VU) *Agent-Based Mediated Service Negotiation*
10 Huib Aldewereld (UU) *Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols*
11 Natalia Stash (TU/e) *Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System*
12 Marcel van Gerven (RUN) *Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty*
13 Rutger Rienks (UT) *Meetings in Smart Environments; Implications of Progressing Technology*
14 Niek Bergboer (UM) *Context-Based Image Analysis*
15 Joyca Lacroix (UM) *NIM: a Situated Computational Memory Model*
16 Davide Grossi (UU) *Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems*
17 Theodore Charitos (UU) *Reasoning with Dynamic Networks in Practice*
18 Bart Orriens (UvT) *On the development and management of adaptive business collaborations*
19 David Levy (UM) *Intimate relationships with artificial partners*
20 Slinger Jansen (UU) *Customer Configuration Updating in a Software Supply Network*
21 Karianne Vermaas (UU) *Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005*
22 Zlatko Zlatev (UT) *Goal-oriented design of value and process models from patterns*
23 Peter Barna (TU/e) *Specification of Application Logic in Web Information Systems*
24 Georgina Ramírez Camps (CWI) *Structural Features in XML Retrieval*
25 Joost Schalken (VU) *Empirical Investigations in Software Process Improvement*

2008

1 Katalin Boer-Sorbán (EUR) *Agent-Based Simulation of Financial Markets: A modular, continuous-time approach*
2 Alexei Sharpanskykh (VU) *On Computer-Aided Methods for Modeling and Analysis of Organizations*
3 Vera Hollink (UvA) *Optimizing hierarchical menus: a usage-based approach*
4 Ander de Keijzer (UT) *Management of Uncertain Data - towards unattended integration*
5 Bela Mutschler (UT) *Modeling and simulating causal dependencies on process-aware information systems from a cost perspective*
6 Arjen Hommersom (RUN) *On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective*
7 Peter van Rosmalen (OU) *Supporting the tutor in the design and support of adaptive e-learning*
8 Janneke Bolt (UU) *Bayesian Networks: Aspects of Approximate Inference*
9 Christof van Nimwegen (UU) *The paradox of the guided user: assistance can be counter-effective*
10 Wauter Bosma (UT) *Discourse oriented Summarization*
11 Vera Kartseva (VU) *Designing Controls for Network Organizations: a Value-Based Approach*

12 Jozsef Farkas (RUN) *A Semiotically oriented Cognitive Model of Knowlegde Representation*
13 Caterina Carraciolo (UvA) *Topic Driven Access to Scientific Handbooks*
14 Arthur van Bunningen (UT) *Context-Aware Querying; Better Answers with Less Effort*
15 Martijn van Otterlo (UT) *The Logic of Adaptive Behavior: Knowledge Representation and Algorithms for the Markov Decision Process Framework in First-Order Domains*
16 Henriette van Vugt (VU) *Embodied Agents from a User's Perspective*
17 Martin Op't Land (TUD) *Applying Architecture and Ontology to the Splitting and Allying of Enterprises*
18 Guido de Croon (UM) *Adaptive Active Vision*
19 Henning Rode (UT) *From document to entity retrieval: improving precision and performance of focused text search*
20 Rex Arendsen (UvA) *Geen bericht, goed bericht. Een onderzoek naar de effecten van de introductie van elektronisch berichtenverkeer met een overheid op de administratieve lasten van bedrijven*
21 Krisztian Balog (UvA) *People search in the enterprise*
22 Henk Koning (UU) *Communication of IT-architecture*
23 Stefan Visscher (UU) *Bayesian network models for the management of ventilator-associated pneumonia*
24 Zharko Aleksovski (VU) *Using background knowledge in ontology matching*
25 Geert Jonker (UU) *Efficient and Equitable exchange in air traffic management plan repair using spender-signed currency*
26 Marijn Huijbregts (UT) *Segmentation, diarization and speech transcription: surprise data unraveled*
27 Hubert Vogten (OU) *Design and implementation strategies for IMS learning design*
28 Ildiko Flesh (RUN) *On the use of independence relations in Bayesian networks*
29 Dennis Reidsma (UT) *Annotations and subjective machines- Of annotators, embodied agents, users, and other humans*
30 Wouter van Atteveldt (VU) *Semantic network analysis: techniques for extracting, representing and querying media content*
31 Loes Braun (UM) *Pro-active medical information retrieval*
32 Trung B. Hui (UT) *Toward affective dialogue management using partially observable markov decision processes*
33 Frank Terpstra (UvA) *Scientific workflow design; theoretical and practical issues*
34 Jeroen de Knijf (UU) *Studies in Frequent Tree Mining*
35 Benjamin Torben-Nielsen (UvT) *Dendritic morphology: function shapes structure*

2009

1 Rasa Jurgelenaite (RUN) *Symmetric Causal Independence Models*
2 Willem Robert van Hage (VU) *Evaluating Ontology-Alignment Techniques*
3 Hans Stol (UvT) *A Framework for Evidence-based Policy Making Using IT*
4 Josephine Nabukenya (RUN) *Improving the Quality of Organisational Policy Making using Collaboration Engineering*
5 Sietse Overbeek (RUN) *Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality*
6 Muhammad Subianto (UU) *Understanding Classification*
7 Ronald Poppe (UT) *Discriminative Vision-Based Recovery and Recognition of Human Motion*
8 Volker Nannen (VU) *Evolutionary Agent-Based Policy Analysis in Dynamic Environments*
9 Benjamin Kanagwa (RUN) *Design, Discovery and Construction of Service-oriented Systems*
10 Jan Wielemaker (UVA) *Logic programming for knowledge-intensive interactive applications*
11 Alexander Boer (UVA) *Legal Theory, Sources of Law & the Semantic Web*
12 Peter Massuthe (TUE, Humboldt-Universtät zu Berlin) *Operating Guidelines for Services*
13 Steven de Jong (UM) *Fairness in Multi-Agent Systems*
14 Maksym Korotkiy (VU) *From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA)*
15 Rinke Hoekstra (UVA) *Ontology Representation - Design Patterns and Ontologies that Make Sense*
16 Fritz Reul (UvT) *New Architectures in Computer Chess*

17 Laurens van der Maaten (UvT) *Feature Extraction from Visual Data*

18 Fabian Groffen (CWI) *Armada, An Evolving Database System*

19 Valentin Robu (CWI) *Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets*

20 Bob van der Vecht (UU) *Adjustable Autonomy: Controling Influences on Decision Making*

21 Stijn Vanderlooy (UM) *Ranking and Reliable Classification*

22 Pavel Serdyukov (UT) *Search For Expertise: Going beyond direct evidence*

23 Peter Hofgesang (VU) *Modelling Web Usage in a Changing Environment*

24 Annerieke Heuvelink (VU) *Cognitive Models for Training Simulations*

25 Alex van Ballegooij (CWI) *RAM: Array Database Management through Relational Mapping*

26 Fernando Koch (UU) *An Agent-Based Model for the Development of Intelligent Mobile Services*

27 Christian Glahn (OU) *Contextual Support of social Engagement and Reflection on the Web*

28 Sander Evers (UT) *Sensor Data Management with Probabilistic Models*

29 Stanislav Pokraev (UT) *Model-Driven Semantic Integration of Service-Oriented Applications*

30 Marcin Zukowski (CWI) *Balancing vectorized query execution with bandwidth-optimized storage*

31 Sofiya Katrenko (UVA) *A Closer Look at Learning Relations from Text*

32 Rik Farenhorst and Remco de Boer (VU) *Architectural Knowledge Management: Supporting Architects and Auditors*

33 Khiet Truong (UT) *How Does Real Affect Affect Affect Recognition In Speech?*

34 Inge van de Weerd (UU) *Advancing in Software Product Management: An Incremental Method Engineering Approach*

35 Wouter Koelewijn (UL) *Privacy en Politiegegevens; Over geautomatiseerde normatieve informatie-uitwisseling*

36 Marco Kalz (OU) *Placement Support for Learners in Learning Networks*

37 Hendrik Drachsler (OU) *Navigation Support for Learners in Informal Learning Networks*

38 Riina Vuorikari (OU) *Tags and self-organisation: a metadata ecology for learning resources in a multilingual context*

39 Christian Stahl (TUE) *Service Substitution – A Behavioral Approach Based on Petri Nets*

40 Stephan Raaijmakers (UvT) *Multinomial Language Learning: Investigations into the Geometry of Language*

41 Igor Berezhnyy (UvT) *Digital Analysis of Paintings*

42 Toine Bogers (UvT) *Recommender Systems for Social Bookmarking*

43 Virginia Nunes Leal Franqueira (UT) *Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Ambients*

44 Roberto Santana Tapia (UT) *Assessing Business-IT Alignment in Networked Organizations*

45 Jilles Vreeken (UU) *Making Pattern Mining Useful*

46 Loredana Afanasiev (UvA) *Querying XML: Benchmarks and Recursion*

2010

1 Matthijs van Leeuwen (UU) *Patterns that Matter*

2 Ingo Wassink (UT) *Work flows in Life Science*

3 Joost Geurts (CWI) *A Document Engineering Model and Processing Framework for Multimedia documents*

4 Olga Kulyk (UT) *Do You Know What I Know? Situational Awareness of Co-located Teams in Multidisplay Environments*

5 Claudia Hauff (UT) *Predicting the Effectiveness of Queries and Retrieval Systems*

6 Sander Bakkes (UvT) *Rapid Adaptation of Video Game AI*

7 Wim Fikkert (UT) *Gesture interaction at a Distance*

8 Krzysztof Siewicz (UL) *Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and eGovernments*

9 Hugo Kielman (UL) *Politiële gegevensverwerking en Privacy, Naar een effectieve waarborging*

10 Rebecca Ong (UL) *Mobile Communication and Protection of Children*

11 Adriaan Ter Mors (TUD) *The world according to MARP: Multi-Agent Route Planning*

12 Susan van den Braak (UU) *Sensemaking software for crime analysis*
13 Gianluigi Folino (RUN) *High Performance Data Mining using Bio-inspired techniques*
14 Sander van Splunter (VU) *Automated Web Service Reconfiguration*
15 Lianne Bodenstaff (UT) *Managing Dependency Relations in Inter-Organizational Models*
16 Sicco Verwer (TUD) *Efficient Identification of Timed Automata, theory and practice*
17 Spyros Kotoulas (VU) *Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications*
18 Charlotte Gerritsen (VU) *Caught in the Act: Investigating Crime by Agent-Based Simulation*
19 Henriette Cramer (UvA) *People's Responses to Autonomous and Adaptive Systems*
20 Ivo Swartjes (UT) *Whose Story Is It Anyway? How Improv Informs Agency and Authorship of Emergent Narrative*
21 Harold van Heerde (UT) *Privacy-aware data management by means of data degradation*
22 Michiel Hildebrand (CWI) *End-user Support for Access to Heterogeneous Linked Data*
23 Bas Steunebrink (UU) *The Logical Structure of Emotions*
24 Dmytro Tykhonov (TUD) *Designing Generic and Efficient Negotiation Strategies*
25 Zulfiqar Ali Memon (VU) *Modelling Human-Awareness for Ambient Agents: A Human Mindreading Perspective*
26 Ying Zhang (CWI) *XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines*
27 Marten Voulon (UL) *Automatisch contracteren*
28 Arne Koopman (UU) *Characteristic Relational Patterns*
29 Stratos Idreos (CWI) *Database Cracking: Towards Auto-tuning Database Kernels*
30 Marieke van Erp (UvT) *Accessing Natural History - Discoveries in data cleaning, structuring, and retrieval*
31 Victor de Boer (UVA) *Ontology Enrichment from Heterogeneous Sources on the Web*
32 Marcel Hiel (UvT) *An Adaptive Service Oriented Architecture: Automatically solving Interoperability Problems*
33 Robin Aly (UT) *Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval*
34 Teduh Dirgahayu (UT) *Interaction Design in Service Compositions*
35 Dolf Trieschnigg (UT) *Proof of Concept: Concept-based Biomedical Information Retrieval*
36 Jose Janssen (OU) *Paving the Way for Lifelong Learning; Facilitating competence development through a learning path specification*
37 Niels Lohmann (TUE) *Correctness of services and their composition*
38 Dirk Fahland (TUE) *From Scenarios to components*
39 Ghazanfar Farooq Siddiqui (VU) *Integrative modeling of emotions in virtual agents*
40 Mark van Assem (VU) *Converting and Integrating Vocabularies for the Semantic Web*
41 Guillaume Chaslot (UM) *Monte-Carlo Tree Search*
42 Sybren de Kinderen (VU) *Needs-driven service bundling in a multi-supplier setting - the computational e3-service approach*
43 Peter van Kranenburg (UU) *A Computational Approach to Content-Based Retrieval of Folk Song Melodies*
44 Pieter Bellekens (TUE) *An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain*
45 Vasilios Andrikopoulos (UvT) *A theory and model for the evolution of software services*
46 Vincent Pijpers (VU) *e3alignment: Exploring Inter-Organizational Business-ICT Alignment*
47 Chen Li (UT) *Mining Process Model Variants: Challenges, Techniques, Examples*
48 Jahn-Takeshi Saito (UM) *Solving difficult game positions*
49 Bouke Huurnink (UVA) *Search in Audiovisual Broadcast Archives*
50 Alia Khairia Amin (CWI) *Understanding and supporting information seeking tasks in multiple sources*
51 Peter-Paul van Maanen (VU) *Adaptive Support for Human-Computer Teams: Exploring the Use of Cognitive Models of Trust and Attention*
52 Edgar Meij (UVA) *Combining Concepts and Language Models for Information Access*

2011

1 Botond Cseke (RUN) *Variational Algorithms for Bayesian Inference in Latent Gaussian Models*
2 Nick Tinnemeier (UU) *Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language*
3 Jan Martijn van der Werf (TUE) *Compositional Design and Verification of Component-Based Information Systems*
4 Hado Philip van Hasselt (UU) *Insights in Reinforcement Learning; Formal analysis and empirical evaluation of temporal-difference learning algorithms*
5 Bas van de Raadt (VU) *Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline*
6 Yiwen Wang (TUE) *Semantically-Enhanced Recommendations in Cultural Heritage*
7 Yujia Cao (UT) *Multimodal Information Presentation for High Load Human Computer Interaction*
8 Nieske Vergunst (UU) *BDI-based Generation of Robust Task-Oriented Dialogues*
9 Tim de Jong (OU) *Contextualised Mobile Media for Learning*
10 Bart Bogaert (UvT) *Cloud Content Contention*
11 Dhaval Vyas (UT) *Designing for Awareness: An Experience-focused HCI Perspective*
12 Carmen Bratosin (TUE) *Grid Architecture for Distributed Process Mining*
13 Xiaoyu Mao (UvT) *Airport under Control; Multiagent Scheduling for Airport Ground Handling*
14 Milan Lovric (EUR) *Behavioral Finance and Agent-Based Artificial Markets*
15 Marijn Koolen (UVA) *The Meaning of Structure: the Value of Link Evidence for Information Retrieval*
16 Maarten Schadd (UM) *Selective Search in Games of Different Complexity*
17 Jiyin He (UVA) *Exploring Topic Structure: Coherence, Diversity and Relatedness*
18 Mark Ponsen (UM) *Strategic Decision-Making in complex games*
19 Ellen Rusman (OU) *The Mind ' s Eye on Personal Profiles*
20 Qing Gu (VU) *Guiding service-oriented software engineering - A view-based approach*
21 Linda Terlouw (TUD) *Modularization and Specification of Service-Oriented Systems*
22 Junte Zhang (UVA) *System Evaluation of Archival Description and Access*
23 Wouter Weerkamp (UVA) *Finding People and their Utterances in Social Media*
24 Herwin van Welbergen (UT) *Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior*
25 Syed Waqar ul Qounain Jaffry (VU) *Analysis and Validation of Models for Trust Dynamics*
26 Matthijs Aart Pontier (VU) *Virtual Agents for Human Communication - Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots*
27 Aniel Bhulai (VU) *Dynamic website optimization through autonomous management of design patterns*
28 Rianne Kaptein (UVA) *Effective Focused Retrieval by Exploiting Query Context and Document Structure*
29 Faisal Kamiran (TUE) *Discrimination-aware Classification*
30 Egon van den Broek (UT) *Affective Signal Processing (ASP): Unraveling the mystery of emotions*
31 Ludo Waltman (EUR) *Computational and Game-Theoretic Approaches for Modeling Bounded Rationality*
32 Nees-Jan van Eck (EUR) *Methodological Advances in Bibliometric Mapping of Science*
33 Tom van der Weide (UU) *Arguing to Motivate Decisions*
34 Paolo Turrini (UU) *Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations*
35 Maaike Harbers (UU) *Explaining Agent Behavior in Virtual Training*
36 Erik van der Spek (UU) *Experiments in serious game design: a cognitive approach*
37 Adriana Burlutiu (RUN) *Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference*
38 Nyree Lemmens (UM) *Bee-inspired Distributed Optimization*
39 Joost Westra (UU) *Organizing Adaptation using Agents in Serious Games*

40 Viktor Clerc (VU) *Architectural Knowledge Management in Global Software Development*

41 Luan Ibraimi (UT) *Cryptographically Enforced Distributed Data Access Control*

42 Michal Sindlar (UU) *Explaining Behavior through Mental State Attribution*

43 Henk van der Schuur (UU) *Process Improvement through Software Operation Knowledge*

44 Boris Reuderink (UT) *Robust Brain-Computer Interfaces*

45 Herman Stehouwer (UvT) *Statistical Language Models for Alternative Sequence Selection*

TiCC Ph.D. Series

1 Pashiera Barkhuysen. *Audiovisual prosody in interaction*. Promotores: M.G.J. Swerts, E.J. Krahmer. Tilburg, 3 October 2008.

2 Ben Torben-Nielsen. *Dendritic morphology: function shapes structure*. Promotores: H.J. van den Herik, E.O. Postma. Co-promotor: K.P. Tuyls. Tilburg, 3 December 2008.

3 Hans Stol. *A framework for evidence-based policy making using IT*. Promotor: H.J. van den Herik. Tilburg, 21 January 2009.

4 Jeroen Geertzen. *Dialogue act recognition and prediction*. Promotor: H.C. Bunt. Co-promotor: J.M.B. Terken. Tilburg, 11 February 2009.

5 Sander Canisius. *Structured prediction for natural language processing*. Promotores: A.P.J. van den Bosch, W.M.P. Daelemans. Tilburg, 13 February 2009.

6 Fritz Reul. *New Architectures in Computer Chess*. Promotor: H.J. van den Herik. Co-promotor: J.W.H.M. Uiterwijk. Tilburg, 17 June 2009.

7 Laurens van der Maaten. *Feature extraction from visual data*. Promotores: E.O. Postma, H.J. van den Herik. Co-promotor: A.G. Lange. Tilburg, 23 June 2009.

8 Stephan Raaijmakers. *Multinomial Language Learning*. Promotores: W. Daelemans, A.P.J. van den Bosch. Tilburg, December 1, 2009.

9 Igor Berezhnyy. *Digital analysis of paintings*. Promotores: E.O. Postma, H.J. van den Herik. Tilburg, December 7, 2009.

10 Toine Bogers. *Recommender Systems for Social Bookmarking*. Promotor: A.P.J. van den Bosch. Tilburg, December 8, 2009.

11 Sander Bakkes. *Rapid adaptation of video game AI*. Promotor: H.J. van den Herik. Co-promotor: P.H.M. Spronck. Tilburg, March 3, 2010.

12 Maria Mos. *Complex Lexical Items*. Promotor: A.P.J. van den Bosch. Co-promotores: Dr. A. Vermeer, Dr. A. Backus. Tilburg, May 12, 2010.

13 Marieke van Erp. *Accessing Natural History. Discoveries in data cleaning, structuring, and retrieval*. Promotor: A.P.J. van den Bosch. Tilburg, June 30, 2010.

14 Edwin Commandeur. *Implicit Causality and Implicit Consequentiality in Language Comprehension*. Promotores: Prof. dr. L.G.M. Noordman, Prof. dr. W. Vonk. Co-promotor: Dr. R. Cozijn. Tilburg, June 20, 2010.

15 Bart Bogaert. *Cloud Content Contention*. Promotores: Prof. dr. H.J. van den Herik, Prof. dr. E.O. Postma. Tilburg, March 30, 2011.

16 Xiauyo Mao. *Airport under Control*. Promotores: Prof. dr. H.J. van den Herik, Prof. dr. E.O. Postma. Co-promotores: Dr. N. Roos and Dr. A. Salden. Tilburg, May 25, 2011.

17 Olga Petukhova. *Multidimensional Dialogue Modelling*. Promotor: Prof. dr. H. Bunt. Tilburg, September 1, 2011.

18 Lisette Mol. *Language in the Hands*. Promotores: Prof. dr. F. Maes, Prof. dr. E.J. Krahmer, and Prof. dr. M.G.J. Swerts. Tilburg, November 7, 2011.

19 Herman Stehouwer. *Statistical Language Models for Alternative Sequence Selection*. Promotores: A.P.J. van den Bosch, H.J. van den Herik. Co-Promotor: M.M. van Zaanen. Tilburg, 7 December 2011.