Weakly supervised methods for information extraction

Koen Deschacht

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

April 2010

D/2010/7515/47
ISBN 978-94-6018-208-2

Abstract

Information extraction (IE) methods detect and classify structured information in unstructured data sources, such as texts and images. Currently, most automatic IE methods are developed with supervised machine learning algorithms that are trained on large, manually annotated datasets. The ability of machine learning algorithms to combine complimentary and contradicting evidence has proved successfully in a wide range of IE tasks. This approach however also suffers from two important disadvantages. The first and most important disadvantage is that for every new task, or for every new domain, a new training corpus needs to be manually annotated. This manual annotation can require the annotation of several thousands of sentences or images, which seriously increases the cost of developing novel IE methods. A second disadvantage is that for complex IE tasks, even a large training set will contain only a fraction of all the relevant structures in the data, which can seriously limit the accuracy of these methods.

In this thesis we study weakly supervised learning, where we develop IE methods that use only a small set of annotated examples, together with a large set of unannotated examples to achieve a high accuracy. We study two settings: (1) unimodal weakly supervised learning, where annotated texts are augmented with a large corpus of unlabeled texts and (2) multimodal weakly supervised learning, where images or videos are augmented with texts that describe the content of these images or videos.

In the unimodal setting we study two IE tasks that extract information from texts. The first task, word sense disambiguation (WSD) determines for every word in the text the meaning of that word, depending on the context. The second task, semantic role labeling (SRL), determines for every verb in the text the semantic frame expressed by that verb and the words in the sentence that are prominent arguments of this verb. The most important core of our models is a directed Bayesian network.

We consider two families of weakly supervised methods to extend the supervised models. The first family of methods are semi-supervised methods, where we learn the parameters of Bayesian network by employing both labeled and unlabeled data. For this we use directed Bayesian networks, where the structures of the unlabeled examples are represented with hidden variables. The values of these hidden variables are then iteratively estimated by optimizing the predictive quality of the Bayesian network on the unlabeled examples. We show that this method is not suitable for IE on texts because of the violation of the assumptions made by this approach. We then turn to a different family of weakly supervised methods, where we first learn an unsupervised model on the unlabeled examples, and use the statistics learned by this model in a supervised machine learning algorithm. We develop an unsupervised model, the latent words language model

(LWLM), that learns accurate word similarities from a large corpus of unlabeled texts. We show that this model is a good model of natural language, offering better predictive quality of unseen texts than previously proposed state-of-the-art language models. In addition, the learned word similarities can be used successfully to automatically expand words in the annotated training with synonyms, where the correct synonyms are chosen depending on the context. We show that this approach improves both the WSD and SRL classifier. Furthermore the LWLM can be used in a wide range of IE and natural language processing applications.

The second part of this thesis discusses weakly supervised learning in a multimodal setting. We develop IE methods to extract certain types of information from texts that describe an image or video, and use this extracted information as a weak annotation of the image or video. We start by developing a method to predict which entities are present in an image. For this we develop two novel measures. The *salience* measure captures the importance of an entity, depending on the position of that entity in the discourse and in the sentence. The *visualness* measure captures the probability that an entity can be perceived visually. This information is extracted in a novel way from the existing WordNet database. We show that combining these measures results in an accurate prediction of the entities present in the image. We then discuss how this model can be used to learn a mapping from names in the text to faces in the image, and to retrieve images of a certain entity.

We then turn to the automatic annotation of video. We develop an SRL system that annotates a video with the visual verbs and their visual arguments, i.e. actions and arguments that can be observed in the video. The annotations of this system are successfully used to train a classifier that detects and classifies actions in the video. A second system annotates every scene in the video with the location of that scene. This system comprises a multimodal scene cut classifier that combines information from the text and the video, an IE algorithm that extracts possible locations from the text and a novel way to propagate location labels from one scene to another, depending the similarity of the scenes in the textual and visual domain.

All the work performed in this thesis is formally evaluated, by comparing the automatic outputs to the ground truth outputs (in the case of IE classifiers), or, by measuring the perplexity of the model on an unseen test text (in the case of the language models). For several tasks we outperform (e.g. WSD and LWLM) or match (e.g. SRL) the best state-of-the-art models. For other tasks we are the first to formally evaluate our system on these tasks (e.g. annotation of visual entities and annotation of locations), setting a competitive baseline for further research.

Kort overzicht

Informatie extractie (IE) methoden detecteren en classificeren gestructureerde informatie in ongestructureerde bronnen, zoals teksten of afbeeldingen. Momenteel maken de meeste automatische IE methoden gebruik van machine leer algoritmes die worden getraind op grote, manueel geannoteerde datasets. De bekwaamheid van machine leer algoritmes om aanvullende of tegengestelde informatie te combineren is succesvol gebleken voor een grote verzameling van IE taken. Deze aanpak heeft echter ook twee grote nadelen. Het eerste en meest belangrijke nadeel is dat voor elke nieuwe taak een nieuw trainingcorpus moet worden geannoteerd. Deze manuele annotatie omvat mogelijk duizenden zinnen of afbeeldingen, wat de kost van de ontwikkeling van IE methodes sterk doet rijzen. Een tweede nadeel is dat voor complexe IE taken, zelfs een grote dataset maar een fractie van alle structuren zal bevatten die herkend moeten worden. Dit kan de nauwkeurigheid van de IE methodes negatief beïnvloeden.

In deze verhandeling bestuderen we zwak gesuperviseerd leren, waarbij accurate IE methodes getraind worden op een kleine verzameling geannoteerde voorbeelden en een grote verzameling niet geannoteerde voorbeelden. We bestuderen twee gevallen: (1) unimodaal zwak gesuperviseerd leren, waar geannoteerde teksten worden aangevuld met een grote verzameling niet geannoteerde teksten (2) multimodaal zwak gesuperviseerd leren, waar afbeeldingen of video's worden aangevuld met teksten die hun inhoud beschrijven.

Voor het unimodale geval bestuderen we twee IE taken die informatie uit teksten extraheren. De eerste taak is de disambiguatie van ambigue woorden afhankelijk van de context waarin die woorden voorkomen. De tweede taak is het bepalen van het semantische frame voor elk werkwoord, samen met de belangrijkste semantische rollen voor dat werkwoord. De IE algoritmen voor deze twee taken worden ontwikkeld met behulp van gerichte Bayesiaanse netwerken. We beschouwen twee categorieën van zwak gesuperviseerde methoden. De eerste categorie zijn semi-gesuperviseerde methoden die de parameters van de Bayesiaanse netwerken leren aan de hand van geannoteerde en niet geannoteerde voorbeelden. In deze netwerken worden de labels van niet geannoteerde voorbeelden voorgesteld met verborgen variabelen. De waardes van deze variabelen worden iteratief geschat door de voorspellende kwaliteit van het netwerk op de niet geannoteerde voorbeelden te optimaliseren. We tonen aan dat deze categorie van methodes niet geschikt is voor IE uit tekst, omdat de veronderstellingen die deze methoden maken niet gelden. Hierna richten we ons op een tweede categorie van zwak gesuperviseerde methoden, waar eerst een ongesuperviseerd model geleerd wordt met niet geannoteerde voorbeelden, en waar dan de statistieken geleerd door dit model gebruikt worden in een gesuperviseerd machine leer algoritme. We ontwikkelen een nieuw ongesuperviseerd taalmodel, het latente woord taalmodel (LWTM), dat de gelijkenis tussen woorden leert aan de hand van een verzameling niet geannoteerde teksten. We tonen aan dat dit model met een hoge accuraatheid niet eerder geziene teksten kan voorspellen. De geleerde gelijkenissen kunnen gebruikt worden om woorden te expanderen met hun synoniemen, welk zowel het systeem voor disambiguatie als het systeem voor het ontdekken van semantische rollen verbetert. Bovendien is de gebruikte methode algemeen en kan ze gebruikt worden in een grote verzameling andere IE methoden.

Het tweede deel van deze thesis behandelt zwak gesuperviseerd leren voor multimodale datasets. We ontwikkelen IE methoden om bepaalde types van informatie te extraheren uit teksten die de inhoud van afbeeldingen of video's beschrijven. De geëxtraheerde informatie wordt dan gebruikt als een zwakke annotatie van de afbeelding of video. We beginnen met het ontwikkelen van een methode die voorspelt welke entiteiten in een afbeelding aanwezig zijn aan de hand van de tekst die de afbeelding beschrijft. We ontwikkelen hiervoor twee nieuwe heuristieken. De *salience* heuristiek modelleert de belangrijkheid van een entiteit in de tekst, aan de hand van de positie van die entiteit in de gehele tekst en in de zin. De *visualness* heuristiek modelleert de kans dat een entiteit visueel kan worden waargenomen, welke op een nieuwe manier wordt bekomen uit de WordNet database. Deze heuristieken resulteren gecombineerd in een nauwkeurige voorspelling van de aanwezige entiteiten in de afbeelding. We tonen ook hoe dit model gebruikt kan worden om de correspondentie te leren tussen namen in de tekst en gezichten in de afbeelding, en om te zoeken naar afbeeldingen met een bepaalde entiteit.

We breiden deze aanpak uit naar de annotatie van video's. We ontwikkelen een systeem voor het detecteren van visuele semantische rollen van visuele werkwoorden, i.e. acties en argument die geobserveerd kunnen worden in de video. De automatisch ontdekte acties en argumenten worden hierna gebruikt om een systeem te trainen dat deze actie en argument automatisch ontdekt in een video. Een tweede uitbreiding is de automatische annotatie van locaties van scenes in de video. Dit systeem combineert informatie uit de tekst en de video om de video onder te verdelen in scenes, en een IE algoritme om locaties uit de tekst te extraheren. We ontwikkelen ook een nieuwe manier om locatie labels te propageren van één scene naar een andere, afhankelijk van de similariteit van de scenes in het tekstuele en visuele domein.

Al de ontwikkelde systemen in deze verhandeling werden formeel geëvalueerd, door ofwel de automatische uitvoer te vergelijken met de manuele annotatie (voor IE methodes), of door de waarschijnlijkheid van een nieuwe tekst volgens het model te meten (voor de taalmodellen). Voor verschillende taken behalen we betere (e.g. woord disambiguatie en latent woord taalmodel) of gelijklopende resultaten (e.g. semantische rol labelen) dan de beste state-of-the-art systemen. Voor andere taken zijn we de eersten die deze resultaten voor deze taken formeel evalueren (e.g. annotatie van visuele entiteiten en annotatie van locaties) en zetten we hiermee een competitieve standaard voor toekomstig onderzoek.

Acknowledgments

First of all I have to thank my supervisors professor Marie-Francine Moens and professor Danny De Schreye. I thank professor Moens for giving me the opportunity to start my PhD and for all the support and guidance she has given me during these four, long years. I would have had a hard time finding a supervisor who is more motivated to do research or who is better in stimulating her students to undertake ambitious research goals, and was very lucky to experience a most pleasant balance between freedom to pursue my own ideas, and help and council when I got completely stuck. I would also like to thank professor De Schreye for supporting me and our research group in our move to the computer science department and in providing me with insights from someone outside the typical NLP community. Furthermore I would like to thank the members of my jury, professor Carlo Vandecasteele, professor Johan Suykens, professor Luc Van Gool, professor Hendrik Blockeel and professor Eric Gaussier for reading my PhD text and for providing me with advice and many detailed comments.

Secondly I would like to thank all my colleagues, new and old, former and present, to remind me from time to time that there is more to life than research. I enjoyed the drinks and the dinners, the parties, the occasional sports activities and the trips abroad! I wish you all the best with your own PhD's! Thanks also to all my other friends in Leuven and Brasschaat. I hope I will have some more time now to go for a beer, play football or simply take down some walls in the new house.

Thanks to Snezhana, for all her patience with my sleepless nights before paper deadlines and for encouraging me when it got all a bit too hard.

I was also very lucky to meet my new flat-mates, who supported me during these hard, last months of the PhD and introduced me to some nice places in Brussels!

And last but certainly not least I would like to thank my family. My dad and mum because they supported me financially and otherwise during my entire studies. I'm very glad they encouraged all of their children to get a university degree. And thanks to my two sisters and brother for accepting that it was just unintelligible to get why doing all this research was actually fun... I hope that this text can prove to them that their brother did not completely waste his time during these four years!

Formal notation

Recurring mathematical symbols used throughout the text:

$BN = (\mathbf{Nodes}, \mathbf{Arcs})$	Bayesian network consisting of collection of nodes **Nodes** and a collection of arcs **Arcs**
Nodes	nodes in a Bayesian network
Arcs	arcs in a Bayesian network
$Node_i$	node i
Dom_i	domain of node i
$Func_i$	probability mass function of node i
Val_i	value of node i
$\mathbf{w}_{train}$	training text
N_t	length training text
$\mathbf{w}_{test}$	test text
N_u	length test text
$\mathbf{w}_{heldout}$	heldout text
N_h	length heldout text
$\mathbf{D}_A$	annotated text
$\mathbf{D}_U$	text that has not been annotated
K	number of features
w_i	word at position i
$\mathbf{Ftr}_{\mathrm{i}}$	all features of word at position i
$\mathbf{Ftr}_{ji}$	features for word i given verb at position j
Ftr_i^k	k-th feature of word at position i
Ftr_{ji}^k	k-th feature for word i given verb at position j
L_i	label of word at position i
θ	generic symbol for parameters of a model

θ_{semi}	parameters of a model learned with semi-supervised learning
θ_A	parameters of a model learned with supervised learning
$\mathcal{L}(\mathbf{D}_A;\theta)$	likelihood of annotated data given parameters
$Synset_i$	synset for word on position i.
$Pred_j$	predicate of word at position j
r_{ji}	role for word i relative to the verb at position j
$\mathbf{r}_j$	all roles for the verb at position j
$L_j = (Pred_j, \mathbf{r}_j)$	labeling for predicate j , i.e. the label $Pred_j$ of the predicate and the labels $\mathbf{r}_j$ of all roles
$\mathbf{w}_{i-n+1}^{i}$	n-gram $[w_{i-n+1}, ..., w_i]$
h_i	hidden word at position i
$c(\mathbf{w}_{i-n+1}^{i})$	counts of the n-gram $\mathbf{w}_{i-n+1}^{i}$ in the training corpus
$d(c(\mathbf{w}_{i-n+1}^{i}))$	discount factor for counts $c(\mathbf{w}_{i-n+1}^{i})$
$\delta(\mathbf{w}_{i-n}^{i-1})$	dynamic interpolation factor
$\pi(w_{i-1}w_i)$	the number of context the bigram $w_{i-1}w_i$ occurs in
C	collection of all counts from all n-gram in the training corpus
γ	collection of all smoothing parameters
$\alpha(\mathbf{h}_{i-n+1}^{i})$	forward values for n-gram $\mathbf{h}_{i-n+1}^{i}$
$\beta(\mathbf{h}_{i-n+1}^{i})$	backward values for n-gram $\mathbf{h}_{i-n+1}^{i}$
$\gamma(\mathbf{h}_{i-n+1}^{i}, h_j)$	forward values for n-gram $\mathbf{h}_{i-n+1}^{i}$ and hidden value h_j
V	vocabulary

Abbreviations

Recurring abbreviations used throughout the text, in alphabetical order:

ADKN	absolute discounted Kneser-Ney smoothing
BN	Bayesian network
HMM	hidden Markov model
IP	interpolated smoothing
LDA	latent Dirichlet allocation
LWLM	latent words language model
ME	maximum entropy
MEMM	maximum entropy Markov model
NB	naive Bayes
pmf	probability mass function
POS	part-of-speech tag
RDKN	relative discounted Kneser-Ney smoothing
SRL	semantic role labeling
WSD	word sense disambiguation

Contents

List of Figures

List of Tables

Introduction

Chapter 1

Outline

“*Nothing clears up a case so much as stating it to another person.*”
The Memoirs of Sherlock Holmes (1893)

In this chapter we introduce the main topics of this thesis. We first discuss the field of natural language processing (section 1.1) and the field of information extraction (section 1.2). In section 1.3 we will see how information extraction algorithms today are mostly developed with machine learning methods. This approach has however some disadvantages that can be solved with weakly supervised learning. Finally we outline the structure of this thesis in section 1.4.

1.1 Natural language processing

This thesis is situated in the field of *natural language processing* (NLP), which we define as

Definition 1.1 ***Natural language processing***

The automatic analysis, transformation and generation of natural language texts using computer algorithms.

As we will see in the following chapters we will mainly be interested in the *analysis* of natural language. To put this in a larger context, we are actually interested in using an automatic analysis of natural language to solve a specific information need of a specific end-user. Natural language processing is only a small part of the full process of solving an information need. An idealized description of such a process is:

1. An end-user has a specific *information need* that can possibly be satisfied by the use of automated natural language processing from natural language texts.
2. A person familiar with NLP analyzes this information need and specifies a formal *task definition* and a description of the *task corpus*.
3. A NLP expert designs and implements *a computer algorithm* to carry out, up to a certain accuracy, the defined task on the given corpus.
4. The computer algorithm is run on the entire task corpus, producing *automatic outputs* for all texts.

Although this description is very general, it helps to outline the topic of this thesis. We perform a study of step 3: *the design of computer algorithms for the automatic analysis of natural language text.* We thus generally assume that the task definition and corpus are known beforehand. Only on a small number of occasions we will address the other steps involved in this process.

1.2 Information extraction

Natural language processing is a broad discipline that comprises many different tasks. We will only be interested in one of these sub-tasks, *information extraction* (IE). Although this term is commonly used to refer to a number of related tasks, it does not have a commonly agreed definition. In this thesis we use the following definition:

Definition 1.2 ***Information extraction***

The extraction of a predefined structure in natural language using computer algorithms, where elements in the structure have a mapping to individual words or phrases in the text.

The elements that distinguish information extraction from the more general natural language processing are *predefined structure* and *mapping to individual words or phrases.* A different view on information extraction states that a predefined structure with a number of slots is given, and that the goal of an IE method is to find the positions where this structure is present in the text and to find the elements, such as words, phrases or sentences that fill one or more of the slots.

In this thesis we perform experiments with a number of different information extraction tasks. Two methods, word sense disambiguation (chapter 3) and semantic role labeling (chapter 4), are used frequently throughout this thesis and allow us to compare the different methods that are developed on a fixed task and corpus.

1.3 Automatic information extraction methods

Our definition of information extraction states that *a computer algorithm* is used to perform an automatic analysis of texts. It does not state however how this algorithm is designed. In this section we outline some of the methods that have historically been used in IE algorithms.

1.3.1 Historical overview

Performing an automatic analysis of text has been a goal of artificial intelligence researchers from the very beginning of computer science (Jurafsky and Martin, 2008). The first extensively studied information extraction task is without doubt syntactic sentence parsing, that tries to discover the structure of a sentence according to a predefined grammar. Research on this topic started blossoming at the end of the 1950's and beginning of the 1960's with the study of formal language theory, generative syntax and automatic parsing algorithms. These early parsing algorithms (e.g Harris (1962)) used pattern matching and keyword search combined with simple heuristics for reasoning. By the end of the 1960's more formal logical systems were developed: Colmerauer (1970) defined a total precedence context free grammar and used the logic programming language Prolog to implement a deterministic sentence parser. Kay (1980) and Pereira and Warren (1983), used an improved parsing algorithm (chart parsing, a dynamic programming algorithm) but were still limited to deterministic algorithms. An important disadvantage of these methods is that for an ambiguous sentence, multiple parses are found without any indication of which parse is more likely. Another successful example at the time is the SHRDLU program developed by Winograd (1972). The program showed the possibilities of natural language processing for human-computer interaction, allowing the user to give commands to a computer program using complex sentences. It combined a sentence parser, a memory of the previous interactions, and a method for disambiguating ambiguous terms depending on the context. It however relied on a simple deterministic (Lisp) implementation and it was not clear how this method could be extended for usage in a broader, more realistic domain.

A small number of information extraction tasks other than syntactic parsing were studied during this time, such automatic pronoun resolution (Hobbs, 1977) and discourse modeling (Grosz et al., 1977). Also here deterministic programs with some simple heuristics were used.

An important shift in methods used for information extraction occurred with the introduction of large annotated corpora, such as the Penn Treebank (Marcus et al., 1994), the Penn Discourse Treebank (Miltsakaki et al., 2004) and the TimeBank (Pustejovsky et al., 2003). These corpora made it possible to use stochastic methods, which had already been successfully applied to other problems, such as optical character recognition (Bledsoe and Browning, 1959) and speech analysis (Jelinek et al., 1975; Baker, 1975). The beginning of the 20th century saw a wide application of machine learning methods, such as support vector machines (Boser et al., 1992; Vapnik, 1995), maximum entropy machines (Berger et al., 1996) and graphical Bayesian models (Pearl and Shafer, 1988). The new corpora also allowed a comparison of different information extraction algorithms on identical test-corpora, a trend that further intensified with the advent of workshops that perform a double-blind comparison of different systems on an identical test corpus. Examples of these workshops are the Message Understanding Conferences[1] (Grishman and Sundheim, 1996) on the detection of various types of events, the Automatic Context Extraction[2] (Doddington et al., 2004) workshops on the detection of entities, relations and events, the Senseval[3] (Kilgarriff, 1998) and related SemEval workshops on word sense disambiguation, semantic role labeling, identification of logic forms, metonymy resolution and other information extraction tasks, and the shared tasks of the Conference on Computational Natural Language Learning[4] (Stevenson and Carreras, 2009), on clause identification, named entity recognition, semantic role labeling and dependency parsing.

1.3.2 Machine learning methods

Today machine learning algorithms are the dominant method to develop information extraction algorithms. A machine learning algorithm is a computer program that automatically constructs (parts of) the information extraction algorithm on the basis of an annotated training set. The training set is a collection of natural language texts that are annotated with the target output structures. Usually a machine learning method aims to minimize a given error measure that quantifies the difference between the automatic constructed outputs and the manual outputs.

[1]http://www.cs.nyu.edu/cs/faculty/grishman/muc6.html
[2]http://www.itl.nist.gov/iad/mig/tests/ace/
[3]http://www.senseval.org/
[4]http://www.cnts.ua.ac.be/conll2010/

Many different machine learning methods have been developed and applied on information extraction methods, we refer to (Manning and Schütze, 2002; Jurafsky and Martin, 2008) for extensive overviews. In this thesis we focus on machine learning methods that are based on graphical Bayesian models. These models use a graphical representation to represent dependencies between different variables, allowing for the construction of complex models which can easily be used for weakly supervised and unsupervised learning. We will discuss graphical Bayesian models at length in chapter 2.

1.3.3 Weakly supervised learning

A major disadvantage of machine learning methods is the large training set that is necessary to learn accurate automatic information extraction algorithms. The PropBank training corpus for semantic role labeling for example, contains 113.000 verbs for which all semantic roles have been manually annotated. A corpus of this size is necessary for most information extraction methods since natural language is very varied and machine learning methods thus need to learn a mapping for a large number of different inputs. Furthermore, the labour intensive task of creating an annotated training corpus needs to be repeated for every information extraction task, or when a specific information extraction method needs to be applied on a corpus in a different language or domain. This requirement greatly increases the costs for the development of IE algorithms, both in terms of time and money.

A solution to this problem is the use of weakly supervised machine learning methods. We define these methods as

Definition 1.3 ***Weakly supervised machine learning methods***

Weakly supervised machine learning methods are machine learning methods that use a labeled together with an unlabeled corpus to train information extraction methods.

Today, large electronic collections of texts in various languages and domains exist and can typically be obtained at a relative small cost. The most important goal of this thesis is the development of weakly supervised algorithms that allow to create information extraction algorithms with a small annotated training corpus. Ideally, one would need to specify only a handful of examples for a given information extraction task to create an automatic computer algorithm that can solve the task up to a high level of accuracy.

In this thesis we consider two types of weakly supervised learning. The first type is *uni-modal weakly supervised learning*, where we augment a small set of annotated texts with a large corpus of unlabeled texts, and use these to improve accuracy of information extraction methods and reduce the dependency of these methods on large annotated corpora. The second type is *multimodal weakly supervised learning*, where we use supervised information extraction methods to automatic generate descriptions of the content of images and video. These descriptions are then used to train methods that perform an automatic analysis of these images or video. This research is motivated by the observation that frequently, the difficulties faced by automatic methods for image analysis are even greater then these faced by natural language processing methods, because of the large variations in scale, lighting conditions and relative orientation of entities in images.

1.4 Outline thesis

We have introduced the major topics of our research and now describe the structure of this text. This thesis is divided in three main parts that discuss respectively supervised information extraction, uni-modal weakly supervised information extraction and multimodal weakly supervised image annotation.

Before diving in the first part of this text we start in **chapter 2** by describing basic concepts and techniques used throughout this work. We give an extensive introduction to *directed Bayesian networks*, which will be used to develop all our supervised, weakly supervised and unsupervised models. This chapter will also describe in more detail how information extraction methods are typically developed, and give a number of examples of popular information extraction methods that will help to situate the information extraction tasks tackled in this thesis.

In **part I** we discuss *supervised information extraction methods.* This part explains in detail how the texts that need to be labeled are converted to a feature representation, how this representation is included in a directed Bayesian network and how this network is used to generate an automatic labeling. We will apply this approach to two specific information extraction tasks. **Chapter 3** discusses *word sense disambiguation*, where we develop a supervised method to determine the meaning of a word depending of the context of that word and **chapter 4** discusses *semantic role labeling* where we develop a supervised method for the automatic detection and classification of the prominent arguments of a verb.

In **part II** we discuss *uni-modal weakly supervised learning*, where we investigate training methods that combine a labeled training set with an unlabeled set to train accurate information extraction methods without relying on large hand-tagged corpora. In **chapter 5** we discuss a traditional approach to this problem: *semi-supervised learning.* This approach uses a single Bayesian network that to represent both labeled and unlabeled examples. The missing labels of the unlabeled examples are iteratively estimated using Markov chain Monte Carlo sampling techniques. We will see that this approach relies on a specific set of assumptions, and how a violation of these assumptions reduces the performance of the final model.

We then turn to a different approach to weakly supervised learning: use a fully unsupervised model to learn statistics or structures from a large unlabeled corpus and use these statistics or structures in a supervised classifier. We develop the latent words language model in **chapter 6**, which is a novel unsupervised model that learns word similarities from a large set of unlabeled examples. These similarities are learned to optimize the predictive accuracy of this model of unseen texts, and can be successfully used in supervised information extraction methods. We demonstrate this in **chapter** 7 by expanding the supervised models used for word sense disambiguation and semantic role labeling with the learned similarities. We also see in this chapter that this method compares favorably to other weakly supervised methods proposed for semantic role labeling.

In **part III** we turn to the case of *multimodal weakly supervised learning*. In this part we discuss methods that employ information extraction methods to aid the automatic analysis of images and video. In **chapter** 8 we develop the *appearance model* which finds the entities present in an image by analyzing a text describing this image. This model is subsequently used in two applications, to align names in the text with faces in the image, and to perform textual image retrieval.

Chapter 9 deals with the automatic annotation of video. We first focus on the automatic annotation of actions of actors in the video, and apply the previously developed semantic role labeling system to the transcripts of an video series. In a second task we combine information extracted from the transcript with an automatic analysis of the video to discover the different scenes in a video, and to derive the location for every scene.

We conclude this thesis in **chapter 10**, where we summarize the work that was performed, the lessons that were learned in the process, and promising directions for future research.

Chapter 2

Foundations

"*Artificial intelligence has done well in tightly constrained domains. Winograd [...] astonished everyone with the expertise of his blocks-world natural language. Extending this kind of ability to larger worlds has not proved straight- forward, however... The time has come to treat the problems involved as central issues*"

Patrick H. Winston (1976)

In this chapter we introduce the basic concepts and techniques used throughout this thesis. We start by describing a number of *important information extraction tasks* in section 2.1. These examples will help to situate the information extraction tasks tackled later in this thesis. We then proceed by giving an extensive introduction to *directed Bayesian networks* in section 2.2. This framework was used in all our models and we will describe how it describes a probability distribution over a collection of variables and how it can be used for machine learning.

2.1 Information extraction tasks

In the previous chapter we have described information extraction (IE) as the task of extracting a predefined structure in natural language using computer algorithms. In this section we make this description more concrete by describing some popular information extraction tasks (section 2.1.1) and by describing the typical process of developing an information extraction method (section 2.1.2).

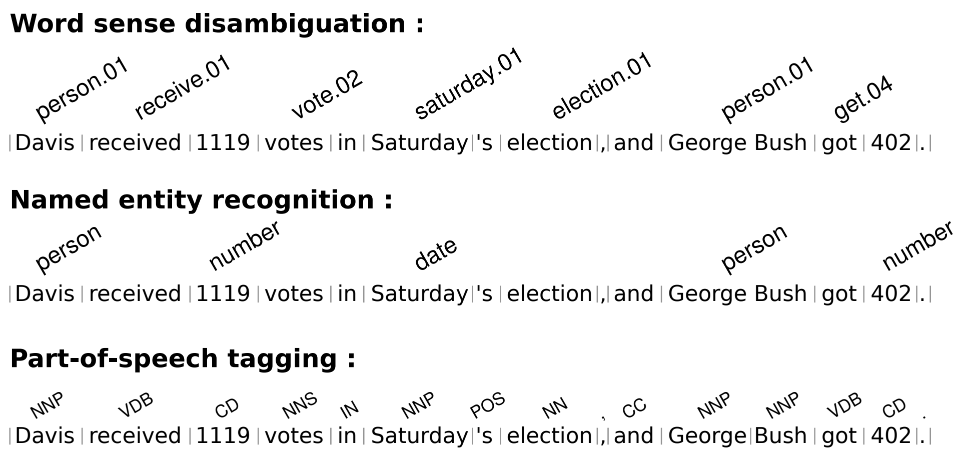

Word sense disambiguation :
person.01
receive.01
vote.02
saturday.01
election.01
person.01
get.04
Davis | received | 1119 | votes | in | Saturday | 's | election | , | and | George Bush | got | 402 | .
Named entity recognition :
person
number
date
person
number
Davis | received | 1119 | votes | in | Saturday | 's | election | , | and | George Bush | got | 402 | .
Part-of-speech tagging :
NNP
VDB
CD
NNS
IN
NNP
POS
NN
,
CC
NNP
NNP
VDB
CD
.
Davis | received | 1119 | votes | in | Saturday | 's | election | , | and | George | Bush | got | 402 | .

Figure 2.1: Example information extraction tasks: word sense disambiguation, named entity recognition and part-of-speech tagging.

2.1.1 Important information extraction tasks

We have previously described how historically syntactic sentence parsing was the first extensively studied IE task. Today a lot of research is still performed on parsing, but other IE tasks are also being extensively researched, of which we describe some in this section. This will help to highlight the shared points in these different tasks and will help to situate the IE tasks studied in this thesis.

Word sense disambiguation The task of word sense disambiguation is to segment a text and to assign a label to every noun phrase, (non-auxiliary) verb phrase, adjective and adverb in a text. This label indicates the meaning for that particular word and is chosen from a dictionary of meanings for a large number of different phrases. For example, in figure 2.1 the word "got" has been assigned the label GET.04 from the WordNet lexical database, indicating that "got" in this context means "receive, obtain, incur".

Named entity recognition Named entity recognition detects names and numbers in a text and classifies these according to a small set of labels, usually including PERSON, ORGANIZATION, LOCATION and DATE. In figure 2.1 both "Davis" and "George Bush" are PERSON names and "Saturday" is a DATE.

Part-of-speech tagging This IE task assigns a syntactic label to every word in a sentence. These syntactic labels reflect the grammatical category of every word in

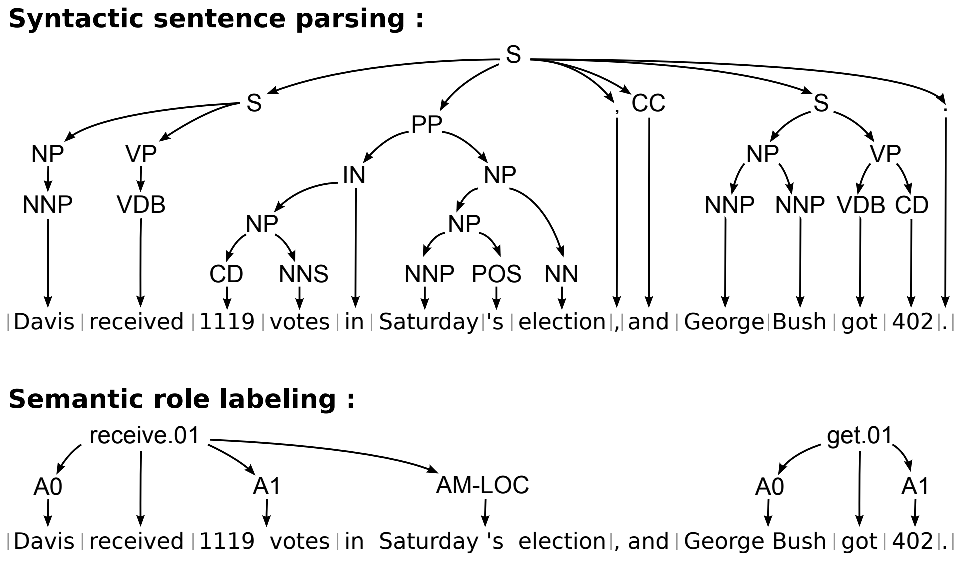


Figure 2.2: Example information extraction tasks: syntactic sentence parsing and semantic role labeling.

the given sentence. In figure 2.1 for example the word “Davis” is assigned “NNP”, “received” is assigned “VBD” and “votes” “NNS”.

Syntactic sentence parsing[1] In syntactic sentence parsing the computer program determines the grammatical structure of a given sentence. Depending on the grammar, this often comes down to finding the syntactic tree for a particular sentence. The words in the sentence are leafs in this tree and internal nodes of the tree are phrases with a syntactic label, i.e. NP for noun phrase or VP for verb phrase. In figure 2.2 the sentence has been parsed using the Penn Treebank constituency grammar (Marcus et al., 1994). It indicates, among others, the part-of-speech tag for every word (e.g. NNP for “Davis”, VDB for “received”) and the structure of the sentence: two independent clauses connected by the conjunction “and”.

[1]In our description of information extraction we do not distinguish between an analysis that focuses on the semantics, i.e. the meaning of words or phrases, an analysis that focuses on the syntactic properties of natural language or an analysis that focuses on the discourse of a particular text. In our experience these different tasks reflect different goals but are very often solved with identical, or very similar techniques. Furthermore there are many tasks (e.g. semantic role labeling, see chapter 4) that straddle these different categories.

Semantic role labeling Semantic role labeling (SRL) annotates every non-auxiliary verb in a sentence with a structure called a semantic frame. A semantic frame consists of a predicate label that indicates the meaning of the verb, and a number of semantic roles. A semantic role is a label for a phrase in the sentence indicating that this phrase is an argument to that verb. In figure 2.2 for example the verb "received" has predicate label RECEIVE.01 with meaning "get,gain" and semantic roles "Davis" with label A0 (receiver), "1119 votes" with label A1 (thing received) and "in Saturday's election" with label AM-LOC (location). Note that the predicate label GET.01 for the verb "get" is different from the synset label GET.04 (figure 2.1) since they are labels from two completely unrelated databases (i.e. PropBank and WordNet).

In this thesis we study word sense disambiguation (WSD) and semantic role labeling (SRL). These two tasks highlight different difficulties for information extraction tasks. The output of WSD is a single label for every word and the difficulty of this task lies mainly in the correct selection of the label, where labels have fine-grained distinctions in meaning that have to be determined from the context. WSD methods thus often focus on creating an accurate model of the context and on methods that are able to learn from only a small number of examples per word sense. SRL in contrast involves selecting a label for a phrase from a very small number of labels. Contrary, the difficulty here lies in the fact that these labels need to be structured in a semantic frame, and that a single word can be used simultaneously in different structures.

2.1.2 Developing information extraction methods

Two points in the design of automatic information extraction methods have generally been the focus of attention: feature extraction and selecting an appropriate machine learning method.

2.1.2.1 Feature extracting

The described information extraction tasks are generally very easy to perform by humans but are extremely hard for computer programs. One of the main reasons is the representation used to store texts. Where humans have an intuitive understanding which texts express similar events, the computer stores texts as linear sequences of characters, where texts with a similar meaning can have completely different representations, or where only a small change to the characters can drastically alter the meaning.

For this reason one of the most important steps in developing information extraction methods is the mapping of the initial representation of texts to a representation that is more useful to the task at hand. This step is usually referred to as feature extraction, and it involves the creation of a set of deterministic rules that look for certain patterns (e.g. suffixes) in the text. Often one uses the output of one information extraction method to create features for a second method. We will for example use the output of a part-of-speech tagger in our word sense disambiguation and semantic role labeling methods. Once the texts are converted to the feature representation, they are passed to a machine learning method.

2.1.2.2 Machine learning methods

The task of a machine learning method is to combine the different values of the various features into a single prediction of the value of the label for that word or phrase. Many different machine learning methods have been applied with success to information extraction, such as support vector machines (Pradhan et al., 2004), neural networks (Collobert and Weston, 2008) k-nearest neighbour classifiers (Morante et al., 2008) decision trees (McCarthy and Lehnert, 1995) or logistic regression (Snow et al., 2005). In this thesis we limit ourselves to a machine learning method based on directed Bayesian networks. This framework is among the most popular methods for information extraction, and it can easily be extended to semi-supervised and unsupervised learning, a fact that is used extensively in the following chapters.

2.2 Directed Bayesian networks

In this section we introduce the framework that is used to develop information extraction algorithms throughout this thesis: directed Bayesian networks. We will formally describe these models (section 2.2.1) and see how we can use them in a machine learning setting (section 2.2.2). Finally we present a famous type of Bayesian networks, hidden Markov models (section 2.2.3), which will be used at several occasions in this thesis.

We only give an introduction to the aspects of Bayesian networks that are relevant to the presented work, for a more complete treatment we refer to Bishop (2006).

2.2.1 Description of graphical Bayesian networks

A Bayesian network is a directed acyclic graph that represents probabilistic dependencies between random variables. A Bayesian network $BN = (\mathbf{Nodes}, \mathbf{Arcs})$ consists of a collection of nodes $\mathbf{Nodes} = [Node_1 ... Node_N]$ and a collection of

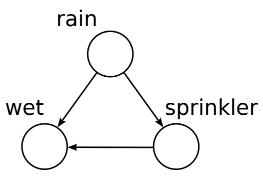


Figure 2.3: Simple Bayesian network

arcs $\mathbf{Arcs} = [Arc_1 ... Arc_M]$. Every node $Node_i = (Val_i, Dom_i, Func_i)$ represents a random variable that can hold a value $Val_i \in Dom_i$ from the domain of that node and a probability mass function $Func_i$ that gives a probability distribution on the domain Dom_i, given the values of the parent of this node. Throughout this text we will use both *variable* and *node* to denote the same concept, a node in the Bayesian network, and often we will use $Node_i = x$ as shorthand for $Val_i = x$ where $x \in Dom_i$.

The collection of arcs **Arcs** of the network capture the dependencies between the nodes in the Bayesian network. $Arc_i = (Node_j, Node_k)$ indicates a directed link from $Node_j$ to $Node_k$. By definition, the model represents a factorization of the joint probability of all random variables.

Definition 2.4 *The probability distribution of a directed Bayesian network* $BN = (\mathbf{Nodes}, \mathbf{Arcs})$ *is given by*

$$P(Node_1, ..., Node_N) = \prod_{i=1}^{N} P(Node_i \mid Parents(Node_i))$$

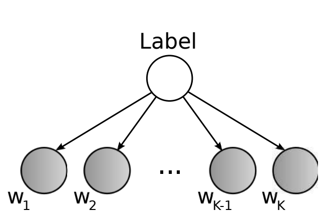
where $Parents(Node_i)$ *denotes the parents of node* $Node_i$

$$Parents(Node_i) = \{Node_j | (Node_j, Node_i) \in \mathbf{Arcs}\}$$

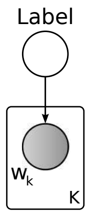
The fact that BN's are directed acyclic graphs, ensures that this decomposition exists and is unique. As an example we consider the small Bayesian network in figure 2.3. This network has three nodes *rainy*, *sprinkler*, and *wet* and three arcs $\{(rainy, sprinkler), (rainy, wet), (sprinkler, wet)\}$, and it represents whether the outlook is rainy (yes/no), whether the sprinkler was turned on last night (yes/no) and whether the lawn is wet (yes/no). The probability distribution of this network has the decomposition

$$P(rainy, sprinkler, wet) = P(wet|rainy, sprinkler) \times P(sprinkler|rainy) \times P(rainy)$$

Some networks will have many variables, for instance one variable for every word in a sentence $\mathbf{w} = [w_1 ... w_N]$ (figure 2.4a). To represent these networks more compactly, we only draw one representative node w_n and surround this node with



(a) Without plate notation



(b) With plate notation

Figure 2.4: Plate notation for Bayesian networks

a box, called a plate, labeled with K indicating that there are K nodes of this kind (figure 2.4b).

Another convention that is used in this work, is that when drawing Bayesian networks, a variable of which the value is known (i.e. observed) is indicated with a shaded circle, and a variable that has an unknown (i.e. hidden or unobserved) value is indicated with an empty circle. In figures 2.4a and 2.4b the nodes indicating the words w_k are set to a certain value and are indicated with shaded circles, while the value of *Label* is unknown and indicated with an empty circle.

Probability mass functions The Bayesian network tells us how to decompose a probability distribution of the network into probability mass functions (*pmf*'s) of single nodes[2]. It however doesn't specify the form of these probability functions. A pmf is defined by

Definition 2.5 *A probability mass function is a function* $Func_i(x)$

$$Func_i(x) : Dom_i \rightarrow [0,1] : Val_i \rightarrow P(Val_i = x)$$

with the constraints that $P(Val_i = x) \geq 0$ *and* $\sum_{x \in Dom_i} P(Val_i = x) = 1$.

Examples of frequently used pmf's are the Bernoulli distribution, the Binomial distribution and the Poisson distribution.

Categorical distribution The probability mass function that will be used most often throughout this work is the *categorical distribution*, which is also referred to in literature as a *multinomial distribution* (e.g. Blei et al. (2003)), although these distributions are in fact only equivalent if the number of trials of the multinomial

[2]Note that we limit ourselves to discrete probability functions, for the more general case, see for instance (Bishop, 2006).

distribution is 1. The parameters of the categorical distribution is a vector of values $[p_1...p_N]$, one for every of the N values in Dom_i, with the constraints that $0 \leq p_j \leq 1$ and $\sum_{j=1}^{N} p_j = 1$. The distribution is then given by a mapping from every value $x_j \in Dom_i$ to the probability p_j of observing this value.

$$Func_i(x) : Dom_i \rightarrow [0, 1] : x \rightarrow p_j$$

Other distributions used in this thesis (e.g. the *Dirichlet distribution*) will be introduced when they are used in a particular information extraction method.

Conditional probability distributions Often we are interested how the probability mass function changes depending on the values of the parents of that node, e.g. we want to model the conditional probability distribution.

Let us take for example the node *sprinkler* in figure 2.3. We could use two different categorical pmf's for this node, one (e.g $[0.02, 0.98]$) for when its parent node *rain* has value *yes* and one (e.g. $[0.30, 0.70]$) for when the value of *rain* is *no*, reflecting the fact that only few people use their sprinklers when the outlook is rainy while more people use them when it is not rainy.

In the general case, we have to consider nodes that have multiple parents. We can choose a similar strategy as for the single parent node, where we use a different pmf for every possible combination of values of the parent nodes. E.g. in the example, we could use 4 different categorical pmf's to model the distribution of *wet* given the 4 combinations of values of *rainy* and *sprinkler*. A serious disadvantage of this approach is that usually we want to learn the parameters of these different functions from training examples, and that a different function for every possible combination of values leads to an explosion of parameters. For this reason we will often use a smarter approach: in section 6.1 we interpolate categorical pmf's for different contexts in language modeling, and in section 4.4 we use exponential models to combine information from many features in a discriminative model.

2.2.2 Machine learning with Bayesian networks

The goal of a supervised machine learning method is to learn, from a set of N manually annotated sentences $\mathbf{D}_A = [(\mathbf{w}_1, L_1)...(\mathbf{w}_N, L_N)]$, a mapping from a sentence $\mathbf{w}_i$ to the label L_i. In reality every sentence $\mathbf{w}_i$ is first transformed to a feature vector $\mathbf{Ftr}_\mathrm{i} = [Ftr_i^1, ..., Ftr_i^K]$, where every feature describes a specific property of the words in the sentence. Generally, the feature vector contains 1 if that property is present and a 0 if the property is not present. We use Bayesian networks in this thesis and create networks that contain variables for both the features and for the labels to be recognized. The structure of the

Bayesian network then specifies the probabilistic dependencies between these variables. In this section we will see some concepts that are important with regard to Bayesian networks in a machine learning setting: the maximum likelihood estimate, the maximum probability estimate and the difference between generative and discriminative Bayesian networks.

Maximum likelihood estimate During the training phase, we learn the parameters of our network from a set of labeled examples $\mathbf{D}_A = [(\mathbf{w}_1, L_1)...(\mathbf{w}_N, L_N)]$. This training set contains N annotated examples that are assumed to be independent and identically distributed. We use θ to denote the parameters of the Bayesian network, which is the union of the parameters of the probability mass functions of all the nodes in the network. A number of methods can be used to estimate these parameters, generally distinguishing between methods that assume or do not assume a *prior distribution* on the parameters, and between methods that find a *single maximum* estimate of the parameters or that model the entire *posterior distribution* for the parameters. We will usually use the *maximum likelihood estimate* (MLE), which does not assume a prior distribution and finds a single maximum for the parameters. We define the likelihood function as:

$$\mathcal{L}(\mathbf{D}_A|\theta) = \prod_{i=1}^{N} P(\mathbf{w}_i, \mathbf{Ftr}_\mathrm{i}|\theta)$$

We then find the parameters θ such that this function has a maximal value. This estimate is guaranteed to produce an optimal classifier for this network, given a 0-1 loss function (DeGroot, 1970) (1) if the number of examples is sufficiently large and (2) if it is possible to find parameters θ such that the parametrized joint likelihood of sentences and labellings $P(\mathbf{w}_i, L_i|\theta)$ equals the true joint likelihood $P(\mathbf{w}_i, L_i)$, or more informally, if the model is “correct”. This is expressed in the following assumption

Assumption 2.1 *Correct model assumption*

We can find parameters θ such that $P(\mathbf{w}_i, L_i|\theta) = P(\mathbf{w}_i, L_i)$.

In our applications both assumptions will generally be violated to some extent, but for supervised models, assumption (1) is generally more important. Since the number of samples is always limited, the maximum likelihood estimate might have some unwanted properties, such as assigning zero probability to events that were never observed in the training set. For this reason we will usually modify the MLE, using a prior distribution (e.g. chapter 9) or smoothing techniques (e.g. section 4.3.2). Although assumption (2) is less important for supervised models, we will see in chapter 5 that for semi-supervised models this assumption is critical.

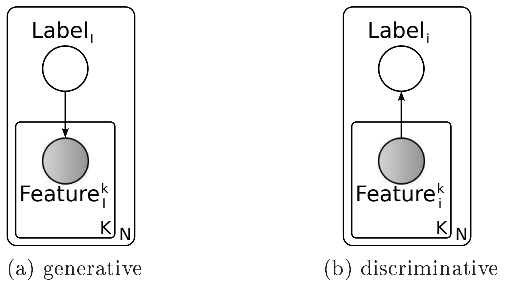


Figure 2.5: Example of a generative and discriminative Bayesian network

Maximum probability estimate After determining the parameters of the Bayesian network, we can use the model to analyze previously unseen texts. For a given text we find the set of values for the unobserved variables (e.g. the labels) such that the joint probability of these variables and of the observed variables (e.g. the words or features) is maximal. A naive approach to find this maximum is to try all possible combinations of variables, which is generally not feasible because of the exponential number of possible combinations. For networks that can be divided in overlapping subnetworks, one can employ a dynamic programming algorithm which is guaranteed to find a global optimal solution (Bellman and Dreyfus, 1962). For other networks one can use a beam search that performs a breadth-first search but only keeps the most likely solutions in every step. This method has the disadvantage that it is not guaranteed to find a global optimal solution.

Generative and discriminative models The number of different Bayesian networks that can be defined is only limited by the creativity of the human mind. However, many tasks have a common setting where a set of observed variables represent the input, and a set of hidden variables represent the labels that need to be assigned to this input. In an information extraction setting for instance, we will represent the texts to be analyzed with a number of features $\mathbf{Ftr}_\mathrm{i} = [Ftr_i^1, ..., Ftr_i^K]$ derived from the word w_i, such as the lemma, prefix or suffix. The (unknown) labels L_i are hidden variables that are assigned to every word w_i. Two possible models to represent the dependencies between the hidden and observed variables are shown in figures 2.5a and 2.5b.

The first model is a generative model, specifying a joint probability distribution over observations (features) and hidden states (labels), or less precise, try to "explain both labels and features". Usually generative networks employ the naive Bayes assumption, where all features are considered independent given the value of the label. The probability distribution of this network is given by

$$P(L_i, \mathbf{Ftr}_\mathrm{i}) = P(L_i) \times \prod_{k=1}^{K} P(Ftr_i^k | L_i)$$

The probability of the labels given the different observed features is

$$P(L_i | \mathbf{Ftr}_\mathrm{i}) = \frac{P(L_i) \times \prod_{k=1}^{K} P(Ftr_i^k | L_i)}{\prod_{k=1}^{K} P(Ftr_i^k)}$$

since $\prod_{k=1}^{K} P(Ftr_i^k)$ is independent of the label, we get

$$P(L_i | \mathbf{Ftr}_\mathrm{i}) \sim P(L_i) \times \prod_{k=1}^{K} P(Ftr_i^k | L_i)$$

A discriminative Bayesian network is shown in figure 2.5b. This network specifies a conditional probability over the hidden states, or less precise, tries only to "explain the labels". The probability distribution of this network is given by

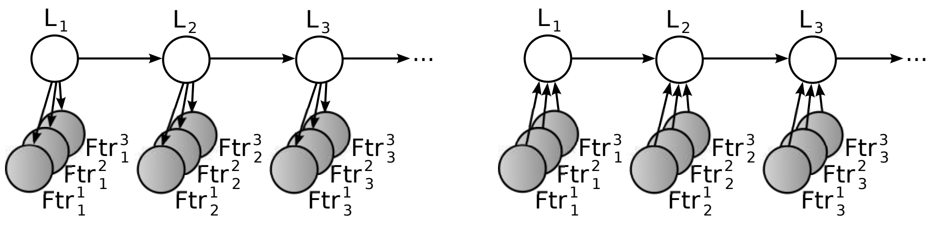
$$P(L_i, \mathbf{Ftr}_\mathrm{i}) = P(L_i | \mathbf{Ftr}_\mathrm{i}) \times \prod_{k=1}^{K} P(Ftr_i^k)$$

These models do not model $P(\mathbf{Ftr}_\mathrm{i})$, since the features are always given, and thus

$$P(L_i, \mathbf{Ftr}_\mathrm{i}) \sim P(L_i | \mathbf{Ftr}_\mathrm{i}) \qquad (2.1)$$

This conditional pmf needs to combine information from many features, which can for example be modeled with an exponential distribution (Ratnaparkhi, 1998).

When comparing the performance of generative and discriminative models, one can compare the asymptotic error, i.e. the error of the model with an unlimited number of labeled examples, and the variance, i.e. the variance of the error when only a limited number of labeled examples is given. Generative models generally use the naive Bayes assumption, considering all features independent given the value of the label, and thus make strong modeling assumptions. For this reason their features can be estimated with a small number of training examples, leading to a low variance with a limited training set. The independence assumption however does usually not hold in practice, which leads to a systematic modeling error even when a large number of labeled examples is observed. Discriminative models with an exponential distribution do generally not have this disadvantage, and will thus generally have a lower asymptotic error. They do however need to model the more



(a) HMM (b) MEMM

Figure 2.6: Hidden Markov model (HMM) and maximum entropy Markov (MEMM) model.

complex conditional distribution in equation 2.1, requiring more training examples. For more information we refer to Bouchard and Triggs (2004) and Ng and Jordan (2002).

For information extraction, discriminative models are usually superior (e.g. for word sense disambiguation (Tratz et al., 2007) or semantic role labeling (Lim et al., 2004)), which is why they are employed here. We will however also consider generative models because of the ease with which they can be used for semi-supervised learning (chapter 5).

2.2.3 Hidden Markov models

One of the more well-known Bayesian networks are hidden Markov models. Hidden Markov models were first used for speech recognition (Baum et al., 1970; Baker, 1975) but have then found a large number of applications, such as natural language modeling (Manning and Schütze, 2002), character recognition (Nag et al., 1986), part-of-speech tagging (Church, 1988; Cutting et al., 1992) and named entity recognition (Bikel et al., 1999). These models have a common structure, shown in figure 2.6. The network consists of a sequence of observed states together with a number of hidden states, and the hidden states are assumed to be dependent on one or more previous hidden states. For example, for speech recognition, the observed states are the spectral vectors of the sound signal during a short period of time and the hidden states are the phonemes or phones, or for part-of-speech tagging the observed states are the words in a text (or features derived from these words) and the hidden states are the part-of-speech tags.

Figure 2.6 shows a HMM where a hidden state h_i has only a conditional dependency with the previous state, a so-called first order HMM. Sometimes however, it can be advantageous to take a larger history into account. This gives rise to second order HMM's, where every hidden state is conditionally dependent on the two previous states, or to a third order HMM where every hidden state is conditionally dependent on the three previous states. Enlarging the history dramatically increases the number of parameters in a model, since the number of parameters is exponential in the number of hidden states. For example, a second order HMM for a part-of-speech tagger with 25 tags will have more than 15625 parameters. We will see in section 6.1 appropriate smoothing methods to overcome the problem of sparseness that are often associated with models with large numbers of parameters.

One of the nice properties of HMM's is that efficient algorithms exist for training and inference. When the hidden states are annotated in the training set (such as for instance when training a supervised part-of-speech tagger), the maximum likelihood parameters of the model can be found trivially in a closed form solution. Also when one tries to learn the hidden states in an unsupervised manner, and they have not been annotated in the training set, one can use an efficient EM-algorithm called the *Baum-Welch algorithm* (Baum et al., 1970). We will discuss this algorithm in detail in section 6.2.2.

Also to find the most likely values for the hidden variables given a sequence of observed variables, an efficient algorithm exists, the Viterbi algorithm (Viterbi, 1967). This algorithm will find the globally maximal sequence of states in a time complexity that is linear in the length of the sequence, and quadratic (for a first-order HMM) or cubic (for a second-order HMM) in the number of possible values for the hidden states.

HMM are by definition generative models, and a number of discriminative models with similar structure and properties have been defined, including *maximum entropy Markov models* (Ratnaparkhi, 1996) and *conditional random fields* (Lafferty et al., 2001). Although these models have been found to have superior performance on a number of information extraction tasks, such as part-of-speech tagging (Lafferty et al., 2001) and named entity recognition (McCallum and Li, 2003), they require significantly more complex methods for training, and can not easily be extended to include hidden variables for semi-supervised training.

Part I

Supervised information extraction

Outline part I : Supervised information extraction

In this part we study the supervised approach to information extraction, which has been the dominant approach to information extraction in the last decade. We first extract a set of features from the word or phrase being labeled, and then use a machine learning method to select he most likely label for this word or phrase. This machine learning method is trained on a large number of manually annotated examples.

When developing a novel information extraction method, the major focus is on selecting the right types of features and on selecting an appropriate machine learning method. In the framework of directed Bayesian networks this comes down to selecting an appropriate structure of the Bayesian network and selecting good probability mass functions. We will demonstrate these techniques on two information extraction tasks: in **chapter 3** we develop a model for word sense disambiguation where the context is modeled with a large number of features, and in **chapter 4** we develop a model for semantic role labeling where we need to model the syntactic role of a word and the relationship between that word to the verb. For both models we will compare a discriminative and a generative Bayesian network.

The work in this part of the thesis has been partially published in the following articles. Parts of this research have not been previously published.

- Koen Deschacht and Marie-Francine Moens. *Efficient Hierarchical Entity Classification Using Conditional Random Fields.* In proceedings of the 2nd Workshop on Ontology Learning and Population, Sydney, 2006.
- Koen Deschacht and Marie-Francine Moens. *Using the Latent Words Language Model for Semi-Supervised Semantic Role Labeling.* In proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP 2009), Singapore, August 7, 2009

A condensed form of the first article was also presented at the 2006 BNAIC conference:

- Koen Deschacht and Marie-Francine Moens. *Efficient Hierarchical Entity Classification Using Conditional Random Fields*, 18th Belgian-Dutch Conference on Artificial Intelligence, Namur, 2006.

Chapter 3

Supervised word sense disambiguation

In this chapter we describe word sense disambiguation, the task of selecting the right sense of a word depending on the context. We introduce this task in section 3.1, and outline the database of word senses and the training and test corpus used in section 3.2. We then develop an automatic method for this task where a number of features to model the context are used in a generative or discriminative Bayesian network in section 3.3. We evaluate these models in section 3.4 and conclude this chapter in section 3.5.

3.1 Introduction

A word can have different meanings depending on the context. Take for example the following sentence

> "The bark sails out of the bay and prepares its cannons for the impending fight."

This sentence contains a number of ambiguous words, such as "bark" (which can mean "sailing ship", "covering of a tree" or "sound made by a dog"), "sails" ("pieces of fabric to propel a sailing vessel" or "to travel on water propelled by wind"), "cannons" ("heavy artillery guns" or "lower parts of the leg in hoofed mammals") and "fight" ("battle" or "boxing or wrestling match"). Although humans can intuitively determine the meanings of the words in this sentence, this is much more complex for computer algorithms. Word sense disambiguation (WSD) is usually defined as *the task of selecting, from a dictionary of possible senses for a particular word, the right sense for a particular word in a particular context.* Generally, one assumes that syntactic disambiguation can successfully be performed with a part-of-speech tagger and that WSD can focus on distinguishing senses among words belonging to the same syntactic category, e.g. finding the correct meaning of "bark", given that it's a noun.

Word sense disambiguation is an important part of natural language processing, since it is a subtask for many tasks, such as machine translation, information retrieval, syntactic parsing, etc. The first research on word sense disambiguation can also be traced to systems that performed machine translation (Weaver, 1955). Since then many WSD systems have been developed, we refer to Ide and Véronis (1998) for an overview. Generally one can distinguish three approaches to WSD, a symbolic approach, a knowledge-driven approach and a data-driven approach. *Symbolic methods* for WSD were usually embedded in larger systems intended for full language understanding. An example is the work by Masterman (1957) who developed a semantic network to derive representations of sentences in an interlingua. In the *knowledge-driven approach* an external knowledge source is used to find the meaning of a particular word in a certain context. A popular knowledge-driven method is for instance to compare the dictionary entries for a particular word with the dictionary entries for words surrounding this word, and select the entry that has the largest number of overlapping words with the entries of surrounding words (Lesk, 1986). In a *data-driven approach*, all ambiguous words in a collection of texts are annotated with the correct sense. A machine learning approach can then be used to learn a classifier for the correct model of the context for a particular sense. An example of early work using this approach is Black (1988).

Today arguably the best comparisons of different WSD systems are the Senseval (Snyder and Palmer, 2004) and SemEval (Pradhan et al., 2007) workshops, where the top ranking systems use machine learning methods that employ large numbers of features extracted from the context (e.g. Tratz et al. (2007)). These features typically capture contextual information (e.g. words surrounding the current word), syntactic information (e.g. subject, object) and semantic information (e.g. named entities types in the context).

3.2 WordNet

We use the WordNet dictionary of senses, which is arguably the most commonly used database for word sense disambiguation. This allows us to compare our work with the best performing systems from the SemEval workshop (Pradhan et al., 2007).

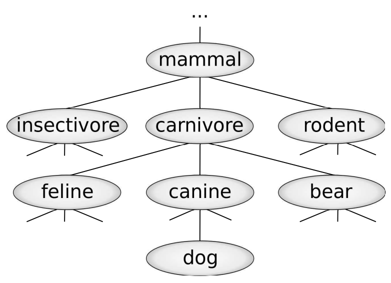


Figure 3.1: Fragment of the WordNet hypernym/hyponym tree

3.2.1 Description

WordNet (Fellbaum, 1998) is a lexical database that organizes English nouns, verbs and adjectives in synsets. A synset is a collection of words that are synonyms, or that are closely related and that represent a single concept or entity. An example of such a synset is "person, individual, someone, somebody, mortal, soul", referring to a human being. The 155.327 words in WordNet (v2.1) are organized in 117.597 synsets. Additionally, WordNet defines a number of relations between synsets, such as the holonym, meronym, pertainym, and the important hypernym relation. A word X is a hypernym of a word Y if Y is a subtype or instance of X. For example, "bird" is a hypernym of "penguin". This relation organizes the synsets in a hierarchical tree of which a fragment is pictured in fig. 3.1.

For a given word, we can list all the synsets that contain this word. The task of WSD using WordNet thus comes down to selecting the correct synset out of all possible synsets for a particular word.

3.2.2 Training and test corpus

We used the Semcor corpus (Fellbaum, 1998; Landes et al., 1998) for training. This corpus, which was created at the Princeton University, is a subset of the English Brown corpus containing almost 700,000 words. Every sentence in the corpus is chunked into noun and verb phrases. The chunks are tagged by part-of-speech and both noun and verb phrases are tagged with their WordNet sense. To be able to compare our system with other systems, we use the test data from the Senseval3 workshop (Snyder and Palmer, 2004), which has been preprocessed in a similar manner.

3.2.3 Evaluation metric

We evaluate our system with the official scorer of the Senseval3 workshop[1]. This scorer measures the accuracy of the assigned labels, i.e.

$$acc = \frac{N_{corr}}{N_u}$$

where N_{corr} is the number of phrases that is assigned the correct label and N_u is the total number of phrases in the test set.

3.3 Supervised WSD models

In this section we discuss the models we have developed for WSD. We first discuss the features employed (section 3.3.1), and then discuss a generative (section 3.3.2) and a discriminative model (section 3.3.3).

3.3.1 Features

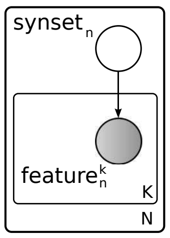
The features used in our system for WSD are mainly based on Tratz et al. (2007).

Contextual information The contextual information we use consists of the word lemmas on either side of the word, within a certain window and within sentence boundaries. Lemmatisation of words, i.e. mapping a word to its canonical form (e.g. “runs”→”run”, “is”→“be”), is performed by an automatic program, part of the WordNet package[2].

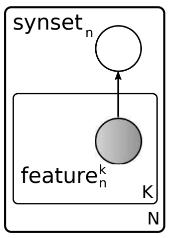
Syntactic information We parse the sentences in their grammatical structure using an automatic parser (Nivre et al., 2006) and include grammatical dependencies (e.g. subject, object) and morpho-syntactic features such as part-of-speech, case, number and tense. Features are extracted for all tokens for which the distance to the word to be disambiguated is smaller then 4 arcs in the dependency tree.

Semantic information We incorporate named entity types (e.g. PERSON, LOCATION, ORGANIZATION). We use OpenNLP and LingPipe to identify named entities, replacing the strings identified as named entities with the corresponding entity type. We also replace numbers in the text with the type label NUMBER.

[1]Available from http://www.senseval.org/senseval3/scoring
[2]Available from http://wordnet.princeton.edu/



(a) generative model



(b) discriminative model

Figure 3.2: Bayesian networks for supervised word sense disambiguation. n is the index of the word out of N words, k the index of the feature out of K features.

Hypernyms Hypernyms are retrieved from WordNet and added to the feature set for all noun tokens selected by the contextual and syntactical rules. We include the hypernyms of the most frequent sense, and we include the entire hypernym chain (e.g. “motor”, “machine”, “device”, “instrumentality”, “artifact”, “object”, “whole” and “entity”).

3.3.2 Generative WSD model

Our first method for WSD uses the network shown in figure 3.2a. This network has two types of nodes, Ftr_i^k and $Synset_i$, representing the features and the synset of a particular word w_i. Categorical pmf's are associated with the $Synset_i$ node and with every individual Ftr_i^k node respectively. The probability distribution of the network is given by

$$P(Synset_i, \mathbf{Ftr}_\mathrm{i}) = P(Synset_i) \times \prod_{k=1}^{K} P(Ftr_i^k | Synset_i)$$

This model is a generative model that assumes that all features are independent, given the synset of the word. The parameters of the categorical distributions are determined using the maximum likelihood estimate, smoothed with a fixed constant. For example, we estimate $P(Ftr_i^k|Synset_i)$ as

$$P(Ftr_i^k|Synset_i) = \frac{c(Ftr_i^k, Synset_i) + \alpha}{c(Synset_i) + \alpha K}$$

where $c(Ftr_i^k, Synset_i)$ is the number of times the feature Ftr_i^k is present in the training set for a word with synset $Synset_i$, $c(Synset_i)$ is the number of occurrences of $Synset_i$ in the training set, α is a positive smoothing constant and K is the number of unique features.

model	nouns	verbs	adjectives	all
generative	57.98	56.43	50.78	55.15
discriminative	65.12	68.15	54.10	66.32

Table 3.1: Results for the generative and discriminative models for WSD on the Senseval3 test set in terms of % accuracy.

3.3.3 Discriminative WSD model

We compare the generative model with a discriminative model, shown in figure 3.2b. Although the network has the same nodes, the direction of the dependency is reversed. The probability distribution is given by

$$\begin{aligned} P(Synset_i, \mathbf{Ftr}_\mathrm{i}) &= P(Synset_i|\mathbf{Ftr}_\mathrm{i}) \times \prod_{k=1}^{K} P(Ftr_i^k) \\ &\sim P(Synset_i|\mathbf{Ftr}_\mathrm{i}) \end{aligned}$$

We model the pmf of $P(Synset_i|\mathbf{Ftr}_\mathrm{i})$ as an exponential distribution, of which the parameters are estimated according to the maximum entropy principle. Since there is no closed form solution to find this maximum we turn to an iterative method. In this work we use generalized iterative scaling[3], although other maximization methods can also be used. Although the maximization method used will have little influence on the final results, more advanced methods (such as (quasi-) Newton optimization) often have a much lower time complexity. The pmf of the features is not modeled, since these probabilities do not influence the relative conditional probabilities of the labels.

3.4 Evaluation of supervised WSD models

For the generative and discriminative models described above we select, from the 5 different types of features described above the combination with a maximal score on a held out set[4]. For the generative model this is the combination of features *Contextual information* and *Syntactic information* and for the discriminative model the combination *Contextual information, Syntactic information*, *Semantic information* and *Hypernyms.* This is in accordance to the common observation

[3]We use the *maxent* package available on http://maxent.sourceforge.net/

[4]To select the best features we train the model on 90% of the Semcor corpus and use 10% to compute the accuracy of the model for a certain combination of features.

that discriminative models are more successful in combining larger number of features compared to generative models. The reason for this is that generative models assume that features are conditionally independent, and a larger number of different types of features are more likely to capture dependencies that violate this assumption.

We then train the model on the Semcor corpus and evaluate on the Senseval3 corpus. We see from the results in table 3.1 that the discriminative model outperforms the generative model, with more than 10% difference in accuracy. Both models perform best for verbs and nouns, and find disambiguation of adjectives particularly hard. This is a trend that is observed for most word sense disambiguation systems. Comparing our results to others, we see that our accuracy is state-of-the-art, slightly higher (Decadt et al., 2004; Kohomban and Lee, 2005; Mihalcea and Faruque, 2004) or lower (Tratz et al., 2007) than others.

It is intriguing that no systems seems to achieve more than 70% accuracy on this dataset. The reason for this is the very fine-grained distinctions in meaning between different WordNet senses. Take for example the noun "man". This noun has 11 different senses, of which three are "the generic use of the word to refer to any human being", "all of the living human inhabitants of the earth" and "any living or extinct member of the family Hominidae characterized by superior intelligence, articulate speech, and erect carriage". It is clear that in a given text, selecting the correct synsets from this set is a non-trivial task, even for humans. For this reason some people have proposed to merge WordNet synsets that are very close in meaning, creating so-called super-senses, which allow automatic systems to achieve much higher accuracies. We have not pursued this approach here.

3.5 Conclusions of this chapter

We have presented a supervised approach to word sense disambiguation. We use a number of features to model the context of a particular word within a certain window, where we use the words in the context, syntactic information, named entities and hypernyms of the words occurring in this window. These features were used in two supervised classifiers, a generative and a discriminative classifier. For both classifiers we found the optimal combination of features, and noticed that the discriminative model could combine a larger number of non-independent features, which can be explained by the independence assumption made by generative models. Upon evaluation on the test set we found that the discriminative model outperforms the generative model, and that its performance is very close to the state-of-the-art.

Chapter 4

Supervised semantic role labeling

In this chapter we address the second information extraction task that will serve as a benchmark of the herein proposed information extraction methods: *semantic role labeling* (SRL). This information extraction task has a long history in linguistics, which we will briefly discuss in section 4.1. We use the PropBank definitions of semantic roles, which we introduce in section 4.2, and in section 4.3 we discuss the features and the two models that are used to tackle this task. We evaluate these models in section 4.4 and conclude this chapter in section 4.5.

4.1 Background

Semantic roles and *semantic frames* have a long tradition in linguistics (Fillmore, 1968; Gruber, 1970) where semantic frames are often defined as script-like structures of common actions or situations and semantic roles as typical participants of, or arguments for, these actions or situations. Historically, semantic frames were proposed as a fundamental building block used by people to organize their memory and conceive the world, e.g. Schank and Abelson (1977).

Today, semantic role labels are usually interpreted as annotations of sentence constituents (e.g. noun phrases) that classify the meaning of the constituent with regard to a verb in the sentence (e.g. by assigning a label to parts of the sentence that answers “who”, “where”, “when”, ... for a particular verb). The term *semantic frame* is used to refer to the semantic roles for a particular verb together with a classification of the verb according to a set of predefined meanings. This provides an analysis of the sentence that can be situated between a grammatical (e.g. syntactic sentence parsing) and semantic analysis (e.g. word sense disambiguation), and offers a semantic structure that generalizes across different syntactic alternations of expressing identical content (Palmer et al., 2005). These structures have been used in a wide range of applications, such as in detecting and filling templates from texts that describe market fluctuations (Surdeanu et al., 2003), selecting correct answers to natural language questions (Narayanan and Harabagiu, 2004; Shen and Lapata, 2007), creating a short summary of a set of documents (Melli et al., 2005), translating texts from one language to another (Boas, 2002) detecting subjective verbs and their arguments (Bethard et al., 2004) and automatic text-to-scene conversion for traffic accident reports (Johansson et al., 2005).

A number of collections of semantic roles have been defined, differing in underlying theoretical assumptions and goals. Three popular collections are FrameNet (Baker et al., 1998), VerbNet (Levin, 1993) and PropBank (Palmer et al., 2005). An alternative approach to semantic role labeling is the framework developed by Halliday (1994) and implemented by De Busser et al. (2002) and Mehay et al. (2005). PropBank has thus far received the most attention of the NLP community, and is used in our work.

4.2 PropBank

4.2.1 Description

The PropBank project (Palmer et al., 2005) defines for a large collection of verbs a set of predicates that reflect the different senses of the verb. The predicates of the verb "run" for example include *run.01* "operate, proceed", *run.02* "walk quickly" and *run.03* "cost". Every predicate label has a number of roles, where label $A0$ is assigned to the most prominent argument in the sentence ($A1$ for unaccusative verbs) and labels $A1$ to $A5$ are assigned to other salient arguments for that verb (Merlo and van der Plas, 2009). Table 4.1 lists the semantic roles for a selection of verb senses. Although roles are defined for every predicate separately, in reality roles with identical names tend to be syntactically and semantically similar for all predicates, a fact that is exploited to train accurate role classifiers. A small number of arguments is shared among all senses of all verbs, such as temporals (AM-TMP), locatives (AM-LOC) and directionals (AM-DIR).

Additional to the frame definitions, PropBank has annotated a large training corpus containing approximately 113.000 annotated verbs. An example of an annotated sentence is

role	**run.01**	**debate.01**	**shoot.02**
A0	operator	debater	shooter
A1	machine, procedure	thing discussed	corpse
A2	employer	person debated against	gun
A3	co-worker	-	location of wound
A4	instrumental	-	-

Table 4.1: Semantic roles in PropBank for a selection of verb senses: *run.01* "operate, proceed", *debate.01* "to discuss" and *shoot.02* "kill with a gun".

[John $_{A0}$] [breaks $_{break.01}$] [the window $_{A1}$] [with a rock $_{A2}$].

Here *break*.01 is the first sense of the verb "break" with meaning "cause to not be whole". John has the semantic role $A0$ "the breaker", "the window" has role $A1$ "thing broken" and "with a rock" has role $A2$ "instrument". The semantic role labeling is preserved across different syntactic realizations. In, for instance, the annotated sentence "[The window $_{A1}$] [broke $_{break.01}$].", "the window" has a different syntactic position but is also assigned role $A1$. Although we will in generally talk about labeling semantic roles, our systems also perform, and are evaluated on, identification of the correct predicate label for the verb (see section 4.2.3).

A sentence with multiple verbs has a separate role labeling for every verb. In for example the sentence

"Big investment banks refused to step up to the plate."

"Big investment banks" is labeled as $A0$ "entity refusing" for the verb "refused" and as $A1$ "thing moving" for the verb "step". Note that semantic role labeling systems typically assume that a frame is fully expressed in a single sentence and thus do not try to instantiate roles across sentence boundaries.

4.2.2 Corpus

We perform our experiments on a standard corpus for semantic role labeling, used in the CoNLL 2008 shared task (Surdeanu et al., 2008)[1]. The sentences in the corpus are mainly taken from news texts in the English language from the Wall Street Journal (from the Penn Treebank corpus (Marcus et al., 1994)) and a small selection of English texts from 15 different sources, including news texts, non-fictional and fictional stories and book reviews (from the Brown corpus (Francis,

[1]Although the CoNLL 2008 shared task evaluated semantic role labeling on both verb and noun phrases, we limit ourselves to semantic role labeling on verbs.

1964)). The corpus is split into three disjoint parts, for training (39279 sentences), testing[2] (2824 sentences) and a held-out section (1334 sentences). The manual annotations of the sentences are the annotations from the PropBank corpus, but are converted from labels for constituents to labels for head words. For example, the annotated text "[John $_{A0}$] [breaks $_{break.01}$] [the window $_{A1}$] [with a rock $_{A2}$]." in the PropBank corpus is converted to "[John $_{A0}$] [breaks $_{break.01}$] the [window $_{A1}$] [with $_{A2}$] a rock.". For details on this conversion, we refer to (Surdeanu et al., 2008).

4.2.3 Evaluation metric

To evaluate the output of our automatic SRL system we use the evaluation metric used in the CoNLL 2008 shared task (Surdeanu et al., 2008). The evaluation metric counts the number of correct labels, i.e. predicate labels and semantic role labels. A predicate label is considered correct if the label corresponds to the label in the manual annotation for that verb. A role label for a particular word is considered correct if the label corresponds to the label in the manual annotation for that word, independently of the label of the verb for that role. This scoring strategy implies that if a system assigns an incorrect predicate label, it still receives some points for the arguments correctly assigned.

For a given test set, N_{corr} is the number of predicates and role labels that are correctly classified, N_{man} is the number of predicates and role labels in the manual annotation and N_{auto} is the number of predicates and role labels in the automatic annotation. Precision, recall and $F1$-measure are defined as

$$precision = \frac{N_{corr}}{N_{auto}} \qquad recall = \frac{N_{corr}}{N_{man}} \qquad F1 = 2 * \frac{precision * recall}{precision + recall}$$

4.3 Supervised SRL models

We will now discuss the models used for semantic role labeling. We first discuss the features used to represent the input text (section 4.3.1) and then consider two different models, based on a generative (section 4.3.2) and discriminative (section 4.3.3) Bayesian network.

[2]In the shared task the test set is split in a set with sentences from the Wall Street Journal and a set with sentences from the Brown corpus. We perform our experiments on all sentences from both sets.

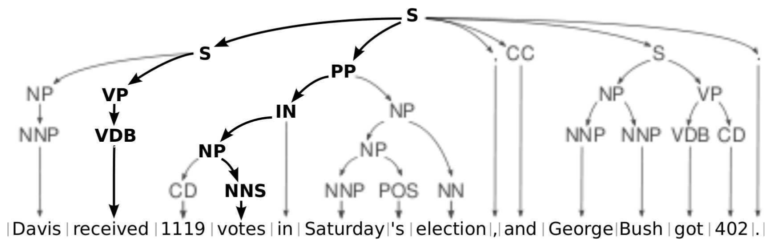


Figure 4.1: Example of the path feature extracted from the syntactic parse tree. The path "nns↑np↑in↑pp↑s↓s↓vp↓vdb" is traversed going from "votes" to "received".

4.3.1 Features

Every word in the CoNLL 2008 corpus is tagged with its part-of-speech by an automatic tagger (Ciaramita and Altun (2006), using the Penn Treebank tags) and a syntactic dependency tree is constructed for every sentence by an automatic parser (Nivre et al., 2006). These automatic annotations are, together with the word tokens, converted to a number of features used in our semantic role labeling classifiers. These features (except *Split path*) have been previously discussed, see for example (Gildea and Jurafsky, 2002; Lim et al., 2004; Thompson et al., 2006). The number in brackets in the following list denotes the number of unique features for that type in the CoNLL 2008 corpus.

Word Unigram word tokens, including punctuation. (37079)

Stem Word tokens reduced to their stem, e.g. "walks" -> "walk". (28690)

POS The part-of-speech tag for every word, e.g. "NNP". (77)

Neighbor POS's The concatenated part-of-speech tags of the word before and the word just after the current word, e.g. "RBS_JJR". (1787)

Path This important feature describes the path through the dependency tree from the current word to the position of the predicate, e.g. "nns↑np↑in↑pp↑s↓s↓vp↓vdb" in figure 4.1, where '↑' indicates going up a constituent and '↓' going down one constituent. (829642)

Split Path Because of the nature of the path feature, an explosion of unique features is found in a given data set. We reduce this by splitting the path in different parts and using every part as a distinct feature. We split, for example, the previous path in 8 different features: "nns", "↑np", "↑in", "↑pp", "↓s", "↓s", "↓vp","↓vdb". Note that the split path feature includes the POS feature, since the first component of the path is the POS tag for the current word. This feature has not been used previously for semantic role detection. (155)

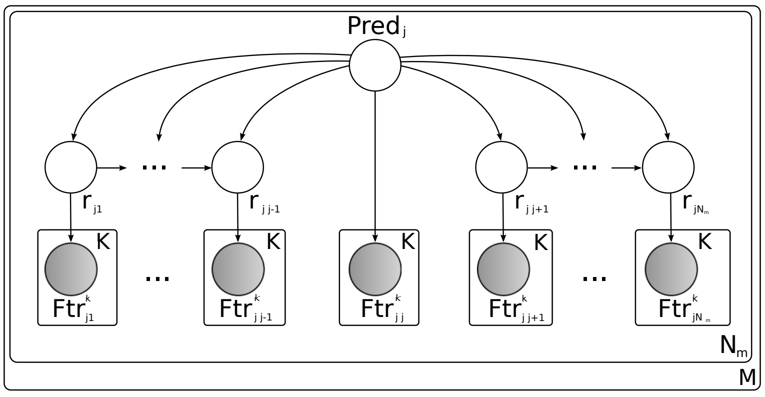


Figure 4.2: Generative model for SRL. m is the index of the sentence, out of a corpus of M sentences, j is the index of the current predicate, out of N_m words in the sentence and k is the index of the feature, out of K features.

Child words The word tokens of the children in the dependency tree of the current word, if any (34518).

Child POS's The POS tag of the children in the dependency tree of the current word, if any (77).

Although most of the described features are independent of the predicate verb, the value of *Path* and *Split_path* differs with regard to the predicate verb. For this reason we use $\mathbf{Ftr}_{ji} = [Ftr^1_{ji}...Ftr^K_{ji}]$ to denote the K features of role r_{ji}, where j indicates the index of the predicate verb.

4.3.2 Generative SRL model

The generative Bayesian network for semantic roles used in this work is very similar to the network proposed in (Thompson et al., 2006) where it is used for semantic frame detection and classification on the FrameNet data set.[3]

For a particular verb w_j at position j in sentence s_m and labeling $L_j = (Pred_j, \mathbf{r}_j)$, where $Pred_j$ is the predicate label and $\mathbf{r}_j = [r_{j1}...r_{jN}]$ are the role labels for all other words in the sentence, the model is defined as in fig. 4.2. The model assumes that the predicate label $Pred_j$ generates the features $\mathbf{Ftr}_{jj}$ and generates

[3]The only major difference is that in (Thompson et al., 2006) the model is not a fully generative model, since the predicate label is generated by the predicate verb. In our model we assume that the predicate label generates the predicate verb. This way the model is a fully generative model that can be used for semi-supervised learning (see chapter 5).

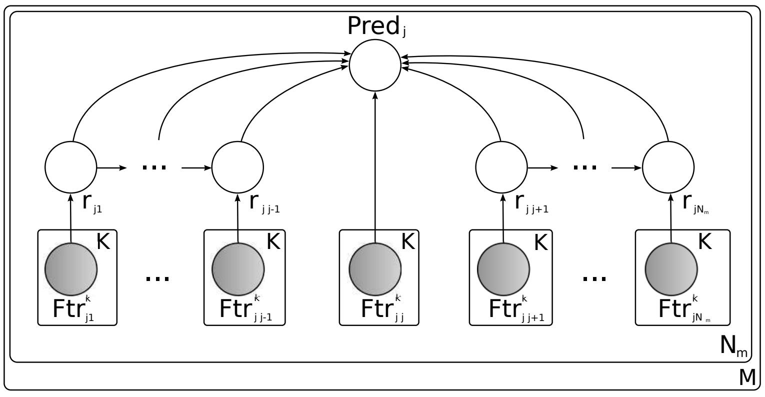


Figure 4.3: Discriminative model for SRL. m is the index of the sentence, out of a corpus of M sentences, j is the index of the current predicate, out of N_m words in the sentence and k is the index of the feature, out of K features.

a sequence of role labels $\mathbf{r}_j$, where every role r_{ji} is dependent on the previous role r_{ji-1}. Finally, every role r_{ji}, $i \neq j$, generates the features $\mathbf{Ftr}_{ji}$.

The probability distribution of this network is given by

$$P(Pred_j, \mathbf{r}_j, \mathbf{Ftr}_{jj}) = P(Pred_j) \times \prod_{i=1}^{N} P(r_{ji}|Pred_j, r_{ji-1})$$

$$\times \prod_{k=1}^{K} P(Ftr_{jj}^{k}|Pred_j) \times \prod_{i=1}^{N} \prod_{k=1}^{K} P(Ftr_{ji}^{k}|r_{ji})$$

We assign a categorical distribution to the $Pred_j$ node, and a collection of categorical distributions to the r_{ji} and Ftr_{ji}^{k} nodes, one for every combination of values of their parents. Given an unlabeled sentence $\mathbf{w}_m = w_1, ..., w_N$ and predicate word w_j, we find the labeling L_j with the highest probability $P(L_j, \mathbf{w}_m)$. Given the predicate label p_j, the model is equivalent to a Hidden Markov Model model, and the optimal labeling L_j can easily be found using the Viterbi algorithm (Viterbi, 1967). To find the optimal predicate label p_{ji}, we run the Viterbi algorithm for every value of the predicate label and find the maximum product of the prior probability of the predicate and the role labels for that predicate.

4.3.3 Discriminative SRL model

Since discriminative models have been found to outperform generative models (Lim et al., 2004) for SRL, we propose an additional discriminative model. The

	5%L	20%L	50%L	100%L
Supervised generative model	38.03%	54.42%	58.38%	68.33%
Supervised discriminative model	40.49%	67.23%	74.93%	78.65%

Table 4.2: Results (in F1-measure) for the fully supervised generative and discriminative models, using different fractions of the CoNLL 2008 training set. Results are average over 10 random subsets.

structure of the model (fig. 4.3(b)) is similar to the previous generative model, although the dependencies have been reversed. The model assumes that the role label r_{ji} for the word w_i is conditioned on the features $\mathbf{Ftr}_{\mathrm{ji}}$ and on the role label r_{ji-1} of the previous word, and that the predicate label $Pred_j$ for word w_j is conditioned on the role labels $\mathbf{r}_j$ and on the features $\mathbf{Ftr}_{ji}$. The likelihood of this model is given by

$$P(Pred_j, \mathbf{r}_j, \mathbf{Ftr}_{j1},) = P(Pred_j|\mathbf{r}_j, \mathbf{Ftr}_{jj}) \times \prod_{i=1}^{N} P(r_{ji}|\mathbf{Ftr}_{ji}, r_{ji-1}, Pred_j)$$

This model can be seen as an extension of the standard maximum entropy Markov model (Ratnaparkhi, 1996) with an extra dependency on the predicate label.

4.4 Evaluation of supervised SRL models

For both classifiers we choose the sets of features used by that classifier as the set that gave best performance when training the classifier on the full training set and testing on a held-out set, disjoint from both training and test set. For the generative model this was the features *Stem*, *Neighbor POS's, Path* and *Child words,* and for the discriminative model, *Word, Stem*, *Neighbor POS's, Split path* and *Child words.*

We perform a number of experiments where we compare the performance of the generative and discriminative models on training sets of different size. We perform experiments with 5%, 20% and 50% and 100% of the full training set. If only a subset of the training examples is used we perform 10 different experiments with random subsets and average the results. A first conclusion that can be drawn from table 4.2 is that the discriminative model outperforms the generative model for all sizes of the training set. This can be attributed to the superior maximum entropy parameter estimation method compared to the maximum likelihood combined with the naive Bayes assumption. Furthermore we see that this difference becomes smaller when training on smaller training sets, suggesting that the NB might prove useful for small training sets. The good performance of the NB classifier on small training sets was also observed by Nigam et al. (1999).

We perform an informal error analysis. Generally speaking errors are caused by two phenomena: ambiguity and underspecification. Ambiguity is common to natural language, and is in this setting mainly caused by words that have multiple meanings (e.g. the verb “run” that can mean “operate” and “walk quickly”) :

> “Mr. Stromach wants to resume a more influential role in *running* the company.”

and

> “[...] insurance generally *runs* a poor second to any direct investment you might make.”

Underspecification occurs when words in the test set (or other new documents) are encountered that have not been seen in the training set. For example, in the following sentence from the test set

> “The dark forms moved like *mourners* on some *nocturnal pilgrimage*, their *dirge unsung* for want of vocal chords.”

the words “mourners” “nocturnal”, “pilgrimage”, “dirge” and “unsung” have not been observed in the training set (nor have their lemmas). In fact, 27.80% of the sentences in the test set of the CoNLL 2008 shared task contain one or more words of which the lemmas are not present in the training set.

4.5 Conclusions of this chapter

In this chapter we have discusses semantic role labeling. We have developed an automatic method for this task that uses a number of features that capture the word, syntactic properties of the word and the syntactic relationship between the word and the predicate verb. We developed a discriminative and a generative classifier and trained these on an annotated corpus. Upon evaluation we found that the discriminative classifier outperforms the generative classifier, which can most likely be attributed to the strong naive Bayes assumption made by the generative classifier. We saw however that for smaller sizes of the training set the performance of the generative model was comparable to that of the discriminative model. Although the discriminative classifier achieves state-of-the-art performance, its accuracy is not fully satisfying. This can be attributed to the ambiguity and sparseness of natural language, and to the fact that often only a limited number of examples is present in the training set for a certain label.

Underspecification and ambiguity are fundamental problems to natural language processing that need to be addressed by every automatic method. Modern machine learning methods have already been a large step forwards compared to manually constructed rules, but still have important limitations. In chapters 5, 6 and 7 we will address the problem of underspecification by augmenting the annotated training set with large amounts of unlabeled data. Additionally the method developed in chapter 6 provides an automatic method for the disambiguation of ambiguous words.

Part II

Weakly supervised information extraction

Outline part II : Weakly supervised information extraction

Directed Bayesian networks can be used to solve some information extraction tasks with a high accuracy. Examples of such tasks are part-of-speech tagging and named entity recognition. For other tasks however this approach does not result in a satisfactory solution. We have seen in the previous chapter, how the models for word sense disambiguation and semantic role labeling achieve only a limited accuracy. This is not only observed for Bayesian networks, but also for other current machine learning methods. The fundamental problem is that a supervised classifier is given by definition only a limited number of annotated examples. Natural language is however very varied, and even a very large training set will only contain a fraction of all possible words and phrases.

In **part II** of this thesis we develop a number of solutions to this problem. In **chapter 5** we focus on *semi-supervised learning*, which is traditionally proposed as a solution to the underspecification problem. Semi-supervised learning uses a data set of labeled and unlabeled examples when training information extraction methods. We study semi-supervised methods based on generative Bayesian networks with hidden variables. We will see how the parameter estimation methods introduced in chapter 2 can easily be extended to the semi-supervised case.

We then propose a different approach to weakly supervised learning: first learn statistics or structures from unlabeled data using an unsupervised model, and in a second step use these statistics or structures as additional information in a supervised model. In **chapter 6** we introduce a novel *unsupervised model*, the latent words language model. This model learns word similarities from a large corpus of unlabeled texts which are used to reduce the sparseness problems related to traditional n-gram models, resulting in a better model of previously unseen texts. In **chapter 7** we will show that these similarities can also be successfully employed in a supervised model for information extraction, resulting in improved performance of the models for word sense disambiguation and semantic role labeling.

The work in this part of the thesis is described in the following articles:

- Koen Deschacht and Marie-Francine Moens. *Using the Latent Words Language Model for Semi-Supervised Semantic Role Labeling.* In proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP 2009), Singapore, August 7, 2010
- Koen Deschacht and Marie-Francine Moens. *The latent words language model.* submitted to Computational Linguistics.
- Koen Deschacht and Marie-Francine Moens. *Weakly supervised learning for semantic role labeling.* submitted to the Journal of Artificial Intelligence Research.

- Koen Deschacht and Marie-Francine Moens. *The Latent Words Language Model.* In Proceedings of the 18th Annual Belgian-Dutch Conference on Machine Learning (Benelearn 09), Tilburg, 2009.

Furthermore the work described in chapter 6 has resulted in the following patent application:

- Koen Deschacht & Marie-Francine Moens. *Method for the automatic determination of context dependent hidden word distributions.* Submitted to U.S. Patent and Trademark Office on November 18, 2009.

Chapter 5

Semi-supervised learning with Bayesian models

“*Data! Data! Data” he cried impatiently. “I can't make bricks without clay.*”
Sherlock Holmes in Doyle (1891)

In this chapter we introduce a first weakly supervised method: semi-supervised learning, which uses both labeled and unlabeled data to train an information extraction method. We introduce semi-supervised learning methods (section 5.1) and describe how generative Bayesian networks can easily be extended to incorporate semi-supervised learning (section 5.2). We then apply a number of variants of this semi-supervised learning method to semantic role labeling in section 5.3, and evaluate these methods in section 5.4. We compare our methods to related research in section 5.5 and summarize our findings in section 5.6.

5.1 Introduction to semi-supervised learning

For most information extraction tasks a large set of examples is needed to learn an accurate mapping from input $\mathbf{s}_i$ to output $\mathbf{L}_i$, requiring a significant investment in terms of time and manual labour. For information extraction from texts this often boils down to manually annotating 10000's of sentences with their correct labeling. Researchers have suggested to alleviate this so-called annotation bottleneck with semi-supervised learning methods that use a set of labeled examples together with a large set of unlabeled examples. A system that could learn an accurate classifier with only a small set of labeled examples and a large set of unlabeled examples (which can typically be collected at a small cost) would substantial reduce the cost of developing IE systems.

A second motivating factor for weakly supervised texts is the inherent *sparseness of natural language texts*. This is for example expressed in Zipf's law (Zipf, 1949; Estoup, 1916), which states that the number of times a word is encountered in any given corpus is inversely proportional to its frequency rank. As a result, most words in a corpus are encountered only a small number of times, and any statistical model needs a way to handle previously unseen words when applied to a new text. Another manifestation of this sparseness is the fact that a sentence of reasonable length has typically never been previously encountered in a given corpus (Katz and Fodor, 1963). A practical result of the sparseness of natural language is that it limits the accuracy of machine learning methods on texts, which is for example expressed in the low accuracy of part-of-speech tagging of previously unseen words (Brants, 2000). Weakly supervised techniques that can analyze today's massive corpora stored on computers with modern day computing power, offer a method to drastically scale up the number of training examples encountered by a given machine learning method, improving its accuracy.

We consider semi-supervised learning methods a subset of the more general weakly supervised learning methods. We define semi-supervised learning as

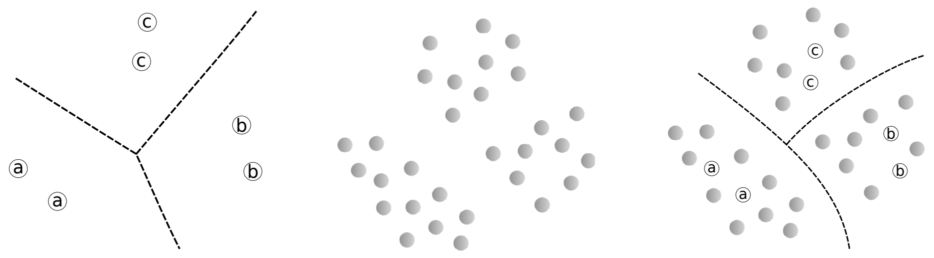
Definition 5.6 ***Semi-supervsed learning methods***

Semi-supervised machine learning methods are methods that optimize a single objective function which incorporates both labeled and unlabeled data.

The most important difference to weakly supervised learning is that a semi-supervised method aims at optimizing a single objective function (e.g. this chapter), while a weakly supervised method can use differernt objective functions to train the parameters from labeled and unlabeled data (e.g. chapter 7).

5.1.1 Assumptions of semi-supervised methods

In this section we explain the assumptions behind most semi-supervised techniques. At the core of most methods are two observations: the first observation is that the space of input examples contains higher density regions (regions containing examples that are likely to be observed) and lower density regions (regions containing examples that are not likely to be observed). The second observation is that low density regions often correlate with classification boundaries. The central idea in semi-supervised learning is to combine these observations: use the unlabeled data to improve the density estimate of the input data and use this



(a) only labeled (b) only unlabeled (c) labeled and unlabeled

Figure 5.1: Schematic figure illustrating how unlabeled data might improve a supervised classifier. Grey dots are unlabeled data, white dots labeled data and the dotted line the classification boundary.

improved density estimate together with the labeled data to learn an improved classification boundary. Figure 5.1 provides a schematic illustration of this idea: a supervised classifier with limited labeled data (fig. 5.1a) is combined with unlabeled data (fig. 5.1b) to learn improved classification boundaries (fig. 5.1c). In a different wording (Chapelle et al., 2006) this assumption can be stated as

Assumption 5.1 *Semi-supervised smoothness assumption*

If two data points s_i and s_j are close in a high-density region, then so should be the corresponding outputs L_i and L_j.

In our work the data points s_i and s_j are words or sentences and the outputs L_i and L_j are labels or labellings that annotate these words or sentences. This assumption has been formulated differently, e.g. as the cluster assumption that states that if two points are in the same cluster they are likely to be of the same class (Seeger, 2002), as the low density separation assumption that states that the decision boundary lies in low density regions of the input space (Chapelle and Zien, 2005), or as the manifold assumption that states that the high-dimensional data lie roughly on a low-dimensional manifold (Belkin et al., 2004). Chapelle et al. (2006) argue that these different formulations can all essentially be interpreted as special cases of the more general smoothness assumption. We will refine this assumption for the specific models used in this work, i.e. semi-supervised (this chapter) and unsupervised (chapter 6) Bayesian models.

5.1.2 Semi-supervised methods

Work on semi-supervised methods in machine learning has been diverse and abundant. Two popular semi-supervised learning methods are *self-training* and *co-training*. These methods can be classified as "meta"-methods, in the sense that they can be employed with any machine learning method. Other methods extend a particular machine learning method, such as transductive support vector machines (Joachims, 1999), the null-category noise model for Gaussian processes (Lawrence and Jordan, 2005), expectation regularization for exponential models (Mann and McCallum, 2007) or generative models with hidden variables (Nigam et al., 2006). We refer to Chapelle et al. (2006) and Zhu (2005) for elaborate literature reviews.

5.2 Semi-supervised learning with generative models

5.2.1 Introduction

In chapter 2 we discussed how the maximum likelihood method is used to compute the parameters of a Bayesian network given a set of labeled training examples. This method can easily be extended to a semi-supervised approach that learns the parameters of the model from a set of labeled and a set of unlabeled examples. Popularized notably by the EM-algorithm (Dempster et al., 1977), this approach has received a lot of attention and its theoretical properties are well understood (Castelli and Cover, 1996; Cozman and Cohen, 2006). We first introduce this method in detail and then apply it to two information extraction tasks.

Given a Bayesian network, a training set $\mathbf{D}_A = [(s_1, L_1)..., (s_a, L_a)]$ of a labeled examples and a set $\mathbf{D}_U = [(s_{a+1}, L_{a+1})..., (s_{a+u}, L_{a+u})]$ of u unlabeled examples. We call the labels of the labeled data $\mathbf{D}_A$ *observed labels*, and the (unknown) labels of the unlabeled data $\mathbf{D}_U$ *hidden labels*, and assume that the probability of a label L_i being observed or hidden is independent of the sample s_i or of the value of the label, i.e. we assume that a random selection of examples were manually annotated.

We now learn the parameters θ_{semi} from these two collections of data by optimizing the combined likelihood of the labeled and unlabeled examples:

$$\mathcal{L}_{semi}(\theta_{semi}) = \sum_{i=1}^{a} log\, P(\mathbf{s}_i, \mathbf{L}_i|\theta_{semi}) + \sum_{i=a+1}^{a+u} log\, P(\mathbf{s}_i|\theta_{semi}) \tag{5.1}$$

where the probability of the unlabeled sentences is computed as

$$P(\mathbf{s}_i|\theta_{semi}) = \sum_{\mathbf{L}_i} P(\mathbf{s}_i|\mathbf{L}_i, \theta_{semi}) \cdot P(\mathbf{L}_i|\theta_{semi})$$

Summarizing, we choose the parameters θ_{semi} so that the model "explains" the labeled examples, by being likely to generate the labellings *and* the sentences, and "explains" the unlabeled examples by being likely to generate the sentences.

This approach builds on the *semi-supervised smoothness assumption* since it assumes that every dense region in the input space can be modeled with a single mixture component. The decision boundaries then lie naturally between these mixture components. A formal treatment of this approach is given by Castelli and Cover (1995), who show that if the Bayesian network is equivalent to the network used to generate the data, and if the mixture components are identifiable (Redner and Walker, 1984), the parameters θ_{semi} can be successfully learned from a collection of labeled and unlabeled examples. Under these assumptions, increasing the size of the unlabeled set increases the accuracy of the parameters, and only a small number of labeled examples is needed to label the mixture components.

Cozman and Cohen (2006) however show that adding unlabeled data can also decrease the performance of the learned model. This can occur if the model used for classification is significantly different from the model that was used to generate the data. They show that violating this *correct model assumption* will in general have only small influence on a fully supervised model, but can potentially dramatically reduce the performance of a semi-supervised model. Furthermore in this case the maximum likelihood estimate will result in different parameters for the fully supervised model than for the unsupervised model. The parameters of the semi-supervised model will then be asymptotically a linear interpolation of these two sets of parameters, and will be closer to the parameters of the supervised or to the parameters of the unsupervised model depending on the ratio of labeled and unlabeled examples.

Semi-supervised learning with generative models on real-world applications has produced mixed results. Positive results have been reported on part-of-speech tagging (Cutting et al., 1992), named entity recognition (Collins and Singer, 1999), face orientation discrimination (Baluja, 1999), and word alignment (Callison-Burch et al., 2004). Negative results were reported on image analysis (Shahshahani and Landgrebe, 1994) and facial expression classification (Grandvalet and Bengio, 2004). Bruce (2001), and Nigam et al. (2000) report mixed results on respectively word sense disambiguation and text classification.

5.2.2 Iterative parameter estimation

To select the variables θ_{semi} given the labeled and unlabeled examples (equation 5.1) we can use two approaches. A first approach is to select the set of parameters for which the likelihood is the highest. This is usually performed with a hill-climbing algorithm that changes the parameters in every iteration such that the likelihood is guaranteed to increase, until a (local) maximum is reached. A popular example of this approach is the EM-algorithm (Dempster et al., 1977). A second method is a Bayesian approach, where a prior distribution is defined for every parameter. From these prior distributions, and the observed examples, the joint posterior distribution for all parameters is computed. The final value of every parameter is then set to the expected value of this parameter according to the posterior distribution, e.g. to the weighted sum of all possible values for this parameter, where the weights are given by the posterior distribution. A popular example of this approach is Markov Chain Monte Carlo sampling (Metropolis and Ulam, 1949). In this chapter we use the latter approach, the former approach will be employed in chapter 6.

Given a set of parameters, a prior distribution for every parameter and a set of examples, we use Markov Chain Monte Carlo (MCMC) sampling to generate samples of these parameters according to the joint posterior distribution. Starting from a random initialization $\mathbf{L}^{(1)}$ of the parameters a Markov chain of samples $\mathbf{L}^{(1)}, ..., \mathbf{L}^{(\tau)}$ is constructed, where the sample $\mathbf{L}^{(\tau)}$ is selected according to a proposal distribution $q(\mathbf{L}^{(\tau)}|\mathbf{L}^{(\tau-1)})$ depending on the previous sample $\mathbf{L}^{(\tau-1)}$. The proposal distribution is chosen such that the chain of samples has the combined likelihood $\mathcal{L}_{semi}$ as the equilibrium distribution. During the first number of iterations (the burn-in period) the samples move from the random start position to the region in the parameter space with high likelihood. After this period, the samples move around the parameter space according to the posterior distribution. Every number of iterations a sample is stored, and at the end of the MCMC method, every parameter is averaged over all collected samples.

5.3 Semi-supervised semantic role labeling

In this section we apply two different MCMC methods to semi-supervised learning for semantic role labeling. We first extend the generative model to semi-supervised learning with Gibbs sampling in section 5.3.1, and propose a new generative model better tailored to semi-supervised learning in section 5.3.2. We also extend the discriminative model with Metropolis-Hastings sampling (section 5.3.3).

5.3.1 Gibbs sampling

We extend the generative model for SRL defined in figure 4.2 to the case of semi-supervised learning. For all labeled sentences we set the predicate and semantic role labels to their manually annotated values. The labels for unlabeled sentences are initially set to a random value[1] and then iteratively updated: in sequence we

[1]A strategy where a classifier was trained on the labeled examples and used to estimate initial values for the unlabeled examples did not result in significant better results.

visit all labels of the unlabeled sentences, every time removing the current label at that position, estimating the probability distribution of the label given the values of all other roles and predicates, and setting a new label randomly selected according to this distribution on this position.

The probability of a new role r_{ji} on position i for verb w_j with predicate $Pred_j$ is given by

$$P(r_{ji}|\mathbf{L}^{(\tau-1)}_{-r_{ji}}, \mathbf{Ftr}_{ji}) \sim \frac{c'(r_{ji-1}, r_{ji}, Pred_j)}{c''(r_{ji-1}, Pred_j)} \times \frac{c'(r_{ji}, r_{ji+1}, Pred_j)}{c''(r_{ji}, Pred_j)} \times \prod_{k=1}^{K} \frac{c'(Ftr^k_{ji}, r_{ji})}{c''(r_{ji})} \tag{5.2}$$

where $\mathbf{L}^{(\tau-1)}_{-r_{ji}}$ is all labellings in iteration $\tau-1$, excluding label r_{ji}, $c'(r_{ji}, r_{ji+1}, Pred_j)$ is the number of times role r_{ji} occurs together with role r_{ji+1} and predicate $Pred_j$ in $\mathbf{L}^{(\tau-1)}_{-r^j_{ji}}$ increased with a pseudo-count α, and $c''(r_{ji}, Pred_j)$ is the number of times role r_{ji} and predicate $Pred_j$ occur in $\mathbf{L}^{(\tau-1)}_{-r_{ji}}$ increased with a pseudo-count $|R|\alpha$ where $|R|$ is the number of distinct values of the role label. c'' is the total number of labels in the dataset increased with $|P|\alpha$, where $|P|$ is the number of distinct values for the predicate label. All other symbols are defined analogously[2]. The pseudo-counts α are the parameters of the prior distribution, a symmetrical Dirichlet distribution. α is chosen to optimize the likelihood of the labels given an unseen test set.

For a new predicate label $Pred_j$ the probability is given by

$$P(Pred_j|\mathbf{L}^{(\tau-1)}_{-Pred_j}, \mathbf{Ftr}_{ji}) \sim \frac{c'(Pred_j)}{c''} \times \prod_{\substack{i=1 \\ i \neq j}}^{N} \frac{c'(r_{ji}, r_{ji+1}, Pred_j)}{c''(r_{ji}, Pred_j)} \times \prod_{k=1}^{K} \frac{c'(Ftr^k_{jj}, Pred_j)}{c''(Pred_j)}$$

A new value is randomly selected for the role or predicate label and the respective variable is assigned the new value. This iteration is performed many times, and after the burn-in period the values of the role labellings are stored at regular intervals and are used to compute the final set of parameters θ_{semi}.

[2]Note that all counts used here can be stored, only to be updated if the value of a role or predicate changes, allowing for an efficient implementation that re-estimates thousands of variables per second.

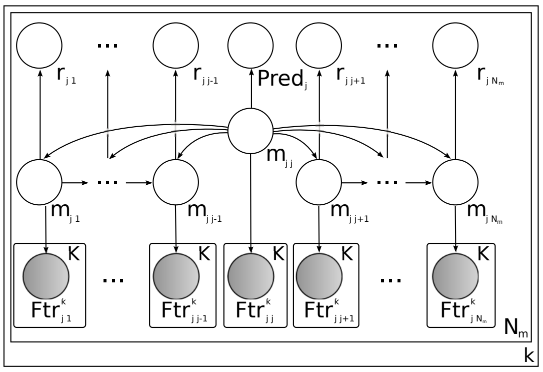


Figure 5.2: Graphical representation of the generative model with multiple mixture components m_i. k ranges over all sentences in the corpus and i over the n words in the sentence.

5.3.2 Gibbs sampling with a multiple-mixture model

We have seen in the previous section how semi-supervised learning is based on the "correct-model" assumption. Although in practice this assumption is almost always violated, this sometimes does and sometimes doesn't decrease performance of semi-supervised methods. If a decrease in performance is observed in semi-supervised models, it can be beneficial to explore a different model that matches more closely the statistics of the data (Cozman et al., 2003). One aspect of the proposed model for semantic role labeling that seems worrisome is the high number of $NULL$ labels. In the training set, more than 91% of the role labels are marked as $NULL$. This does not seem to be a good model of natural language, since it tries to model almost all words with a single label. Nigam et al. (2006) have noted that in the presence of a mismatch between labels and true mixture components, it can be advantageous to define a number of hidden mixture components, with a many-to-many mapping between mixture components and role labels. With this setup, we see a semantic role as a label for a cluster of natural language phenomena (e.g. prepositional phrases expressing a location). This very general cluster however likely consists of a number of sets of sentence constituents that are semantically and syntactic related, i.e. a number of sub-clusters. The mixture components can then be used to represented different sub-clusters, leading to a better representation of the different semantic roles. We expect that this will be especially helpful to obtain a more accurate model for the $NULL$ label.

5.3.2.1 Description of the model

The multiple mixture Bayesian model is defined as in figure 5.2. For a verb w_n at position n in sentence s_m of N words we introduce a vector of N mixture components $\mathbf{m}_i = [m_{i1}, ..., m_{iN}]$. The mixture components are never observed in the training data, and are learned iteratively from the labeled and unlabeled data. We set the number of unique mixture components to 40. Although we assume that, after training, a single mixture component will map to a single role, this constraint is never enforced during sampling. We will see in section 5.4 how this influences our results. We estimate the mixture components from the joint likelihood of the labels and observed features for the labeled examples and of the likelihood of the observed features for the unlabeled examples

$$\mathcal{L}_{semiMM}(\theta_{semi}) = \sum_{k=1}^{a} log\, P(\mathbf{s}_i, \mathbf{L}_i|\theta_{semi}) + \sum_{i=a+1}^{a+u} log\, P(\mathbf{s}_i|\theta_{semi})$$

where the joint probability of a sentence and its labels is computed as

$$P(\mathbf{s}_j, \mathbf{L}_j|\theta_{semi}) = \sum_{\mathbf{m}_{ji}} [P(\mathbf{s}_j, \mathbf{L}_j|\theta_{semi}, \mathbf{m}_{ji}) \cdot P(\mathbf{m}_{ji}|\theta_{semi})]$$

and the probability of an unlabeled sentence is computed as

$$P(\mathbf{s}_j|\theta_{semi}) = \sum_{\mathbf{m}_{ji}} [P(\mathbf{s}_j|\theta_{semi}, \mathbf{m}_{ji}) \cdot P(\mathbf{m}_{ji}|\theta_{semi})]$$

Note that we do not estimate the role labels of the unlabeled examples, since these do not influence the likelihood of the observed features, given the mixture components.

5.3.2.2 Training

We use Gibbs sampling to estimate the parameters of this model: we first set all mixture components to a random value and then sequentially visit every mixture component m_{ji}, remove the current value for that component, and compute a probability distribution for the value of this mixture component given the features, the labels and all other mixture components. The probability distribution of a mixture component in an unlabeled sentence is given by

$$\begin{aligned} P(m_{ji}|\mathbf{M}^{(\tau-1)}_{-m_{ji}}, \mathbf{Ftr_{ji}}, r_{ji}) \quad \sim & \quad \frac{c'(m_{ji-1}, m_{ji+1}, m_{jj})}{c''(m_{ji-1}, m_{jj})} \times \frac{c'(m_{ji}, m_{ji+1}, m_{jj})}{c''(m_{ji}, m_{jj})} \\ & \times \frac{c'(m_{ji}, r_{ji})}{c''(m_{ji})} \times \prod_{k=1}^{K} \frac{c'(Ftr^k_{ji}, m_{ji})}{c''(m_{ji})} \end{aligned}$$

where $M^{(\tau-1)}_{-m_{ji}}$ is the collection of mixture components of all labellings in iteration $\tau - 1$, excluding component m_{ji}. Other symbols in this formula are analogous to symbols used in section 5.3.1. The distribution of a new value of mixture component m_{ji} of a semantic role for an unlabeled sentence is

$$P(m_{ji}|\mathbf{M}^{(\tau-1)}_{-m_{ji}}, \mathbf{Ftr_{ji}}) \quad \sim \quad \frac{c'(m_{ji-1}, m_{ji+1}, m_{jj})}{c''(m_{ji-1}, m_{jj})} \times \frac{c'(m_{ji}, m_{ji+1}, m_{jj})}{c''(m_{ji}, m_{jj})} \times \prod_{k=1}^{K} \frac{c'(Ftr^k_{ji}, m_{ji})}{c''(m_{ji})}$$

We thus see that for an unlabeled sentence, the mixture components are only dependent on the other mixture components and the features, not on the (unknown) labeling.

5.3.2.3 Inference

The joint probability of a labeling $L_j = (Pred_j, \mathbf{r}_j)$ and a sentence $\mathbf{w}$ is given by

$$P(L_j, \mathbf{w}) = \sum_{\mathbf{m}_{ji}} P(\mathbf{w}, L_j|\mathbf{m}_{ji}, \theta_{semi}) \cdot P(\mathbf{m}_{ji}|\theta_{semi})$$

The mixture components are thus marginalized, i.e. summed out. During classification we find the predicate label and roles that maximizes this probability, using a beam search.

5.3.3 Metropolis-Hastings sampling

We have seen in chapter 4 how the discriminative model significantly outperforms the generative model, and we would also like to expand this model to semi-supervised learning. This is however not as straightforward as for the generative model, since in principle unlabeled examples do not influence the conditional likelihood of the labels in a discriminative model (Chapelle et al., 2006). However, we can employ a more general MCMC sampling method, Metropolis-Hastings sampling (Hastings, 1970; Bishop, 2006). In Metropolis-Hastings sampling the sample $\mathbf{L}^{(\tau+1)}$ is selected conditioned on the previous sample $\mathbf{L}^{(\tau)}$ according to some proposal distribution $q(\mathbf{L}^{(\tau+1)}|\mathbf{L}^{(\tau)})$. The sample is then accepted or rejected with a probability given by

$$G(\mathbf{L}^{(\tau+1)}, \mathbf{L}^{(\tau)}) = min\left(1, \frac{\mathcal{L}'_{semi}(\mathbf{S}, \mathbf{L}^{(\tau+1)}) \cdot q(\mathbf{L}^{(\tau)}|\mathbf{L}^{(\tau+1)})}{\mathcal{L}'_{semi}(\mathbf{S}, \mathbf{L}^{(\tau)}) \cdot q(\mathbf{L}^{(\tau+1)}|\mathbf{L}^{(\tau)})}\right)$$

Here $\mathcal{L}'_{semi}(\mathbf{S}, \mathbf{L}^{(\tau)})$ is the likelihood $\mathcal{L}_{semi}(\mathbf{S}, \mathbf{L}^{(\tau)})$ of the sentences $\mathbf{S}$ and labellings $\mathbf{L}^{(\tau)}$ multiplied with some value δ, which can be any non-zero value, as long as it remains constant during the sampling process. We choose $\delta = \frac{1}{\mathcal{L}(\mathbf{S})}$ (i.e. the inverse of the likelihood of the sentences) which is a constant (although unknown) value and see that $\mathcal{L}'_{semi}(\mathbf{S}, \mathbf{L}^{(\tau)}) = \mathcal{L}_{semi}(\mathbf{L}^{(\tau)}|\mathbf{S})$. This is the conditional likelihood of the labellings given the sentences, a value that can be computed with the discriminative model. We set the proposal distribution to $q(\mathbf{L}^{(\tau+1)}|\mathbf{L}^{(\tau)}) = P(\mathbf{L}^{(\tau+1)}|\theta^{(\tau)}_{semi}, \mathbf{S})$, where the parameters $\theta^{(\tau)}_{semi}$ are learned from the previous labeling $\mathbf{L}^{(\tau)}$ and the sentences $\mathbf{S}$ using maximum entropy with generalized iterative scaling. Similarly we define $q(\mathbf{L}^{(\tau)}|\mathbf{L}^{(\tau+1)}) = P(\mathbf{L}^{(\tau)}|\theta^{(\tau+1)}_{semi}, \mathbf{S})$, where the parameters $\theta^{(\tau+1)}_{semi}$ are learned from the labeling $\mathbf{L}^{(\tau+1)}$.

Summarizing, the acceptance function $G(\mathbf{L}^{(\tau+1)}, \mathbf{L}^{(\tau)})$ combines the traditional exploit/explore trade-off. The factor $\frac{\mathcal{L}_{semi}(\mathbf{L}^{(\tau+1)}|\mathbf{S})}{\mathcal{L}_{semi}(\mathbf{L}^{(\tau)}|\mathbf{S})}$ moves the sampling method towards labellings that maximize the conditional likelihood, exploiting known good areas in the sample space, while $\frac{P(\mathbf{L}^{(\tau)}|\theta^{(\tau+1)}_{semi}, \mathbf{S})}{P(\mathbf{L}^{(\tau+1)}|\theta^{(\tau)}_{semi}, \mathbf{S})}$ encourages labellings that are "unlikely" given the current labeling, thereby exploring new areas in the sample space. Also here, after the burn-in period a number of samples are stored which are combined in the final model.

5.4 Evaluation of semi-supervised SRL

We evaluate the sampling methods using 20% of the examples of the CoNLL 2008 training set as labeled examples, and different sizes of the training set as unlabeled examples, ranging from 0% (i.e. a fully supervised classifier) to 80%. A random labeling is created for the unlabeled examples, which is then iteratively updated during the Gibbs or Metroplis-Hastings sampling method. After an initialization period (i.e. the burn-in period) we collect samples every 20th iteration. The collected samples are used to train a final model, which is then evaluated on the CoNLL 2008 test corpus. We report the F1-measure of the different methods in table 5.1. For the Gibbs sampling method, we see that adding unlabeled examples degrades performance, from 54.42% for a classifier that is trained on 20% of all labeled examples and no unlabeled examples, to 43.75% for a classifier that was

	20%L+0%U	20%L+20%U	20%L+40%U	20%L+60%U	20%L+80%U
Gibbs	54.42%	48.61%	47.43%	45.86%	43.75%
Gibbs MM	53.55%	52.39%	49.11%	51.70%	50.91%
M-H	67.23%	62.12%	59.40%	60.68%	59.19%

Table 5.1: Performance (in F1-measure) of the generative model trained with Gibbs sampling, the generative multiple-mixtures model trained with Gibbs sampling and the discriminative model trained with Metropolis-Hastings sampling. The models use 20% labeled (L) and various fractions of unlabeled data (U) from the CoNLL training set.

trained with 20% labeled and 80% unlabeled data. We must thus conclude that for the generative model, the violation of the correct-model assumption is too severe, and adding unlabeled data makes the parameters move away from the optimal parameters for classification, resulting in a dramatically reduced performance.

We have proposed the multiple mixtures model as a model that has more flexibility to model natural language. We see that for 20% labeled data and no unlabeled data, the multiple mixtures model performs slightly worse than the standard generative model. The reason for this is that also for the labeled corpus we need to estimate the mixture components, which might results in ambiguity between some mixture components and labels. We see however that the performance of the multiple-mixture components model is more stable when adding more unlabeled examples, reducing only by less than 3% when using 4 times the number of unlabeled examples compared to the number of labeled examples. These results indicate that the additional degrees of freedom provided by the multiple mixture components allow the model to model natural language more closely, making it better suited for semi-supervised learning. This can also be understood from the *semi-supervised smoothness assumption* which suggests that if the dense regions can be modeled more accurately, labels can be propagated more reliably to unlabeled examples.

Table 5.2 gives the mapping from role mixtures to role labels that was automatically learned during Gibbs sampling (using 20% labeled and no unlabeled examples). We see that some role mixtures have a clear mapping to a single role label, such as mixture 0 to $NULL$ role, mixture 1 to role $A1$ and mixture 3 to $A0$. Other mixtures, such as mixture 5 and mixture 14 are more ambiguous, and map to multiple role labels, which is a likely cause of errors. We also see that many roles map to the $NULL$ role, confirming our hypothesis that this label in fact models different types of sentence constructs, which are better modeled with a larger number of mixtures.

Finally we evaluate the Metropolis-Hastings sampling algorithm. We start from the discriminative model, which significantly outperforms the generative model

	A0	A1	A2	AM-ADV	AM-DIS	AM-LOC	AM-MNR	AM-MOD	AM-TMP	NULL	total
0	0.62	1.38	0.03	0.00	0.00	0.03	0.06	0.03	0.10	96.80	43146
1	1.25	86.20	0.15	0.19	0.01	0.07	0.03	0.08	0.49	10.93	19544
2	0.38	0.58	0.08	0.05	0.00	0.00	0.03	0.04	0.04	97.96	62493
3	92.74	1.16	0.89	0.95	0.12	0.00	0.41	0.08	0.12	3.26	6548
4	0.12	93.77	0.31	0.07	0.02	0.07	0.13	0.07	0.02	4.81	27665
5	0.19	45.54	0.20	0.06	0.05	0.04	0.01	0.07	0.02	53.74	56295
6	0.05	0.09	0.35	0.06	0.04	0.00	0.00	0.00	0.26	98.86	34147
7	0.12	0.05	0.01	0.01	0.01	0.01	0.00	0.01	0.06	99.35	83408
8	0.77	0.09	97.28	0.15	0.17	0.27	0.10	0.01	0.24	0.72	14263
9	0.96	0.70	0.09	0.00	0.09	0.06	0.01	0.22	0.19	97.41	21112
10	0.79	1.37	0.12	0.20	0.21	0.16	0.03	0.63	86.80	9.36	11082
11	96.09	1.03	0.18	0.00	0.02	0.05	0.00	0.01	0.02	2.19	49295
12	0.04	5.51	0.88	0.73	0.31	0.25	0.14	0.62	0.44	90.40	9455
13	0.18	4.52	1.03	12.91	0.08	0.01	16.13	41.89	0.22	22.21	11609
14	24.69	64.73	0.55	0.17	0.17	0.08	0.03	0.12	0.19	8.72	13029
15	0.13	0.03	0.07	0.00	0.02	0.02	0.04	0.02	0.08	98.90	82504
16	2.45	3.43	2.87	0.15	0.00	0.40	0.35	0.59	0.44	88.82	5009
17	10.62	0.78	4.71	0.15	0.02	16.88	1.23	2.66	0.27	61.24	3661
18	1.16	0.56	0.06	0.20	0.14	0.05	0.09	0.22	0.37	96.24	14655
19	3.06	0.79	69.58	0.55	0.18	0.09	0.28	0.05	0.41	24.03	5803

Table 5.2: For the 20 most frequent role mixtures (first column) and the 10 most frequent role labels (first row) in the labeled training set, this table lists the number of assignments of role mixtures to role labels after the Gibbs sampling (using 20% of the labeled training data and no unlabeled data), numbers are given in percentages of total occurrences of the role mixture (last column). For the definition of the various roles, see table 4.1 on page 37.

when using only labeled data. As we add more unlabeled data, we see that also here, the performance of the method decreases, although the decrease (2.93%) is smaller then for the generative model.

5.5 Related work

Concerning research on semi-supervised methods for natural language processing we mention the application of self-learning on word sense disambiguation (Yarowsky, 1995) and on syntactic sentence parsing (McClosky et al., 2006), the use of co-training for syntactic parsing (Sarkar, 2001) and part-of-speech tagging (Clark et al., 2003), semi-supervised discriminative methods for detecting and labeling gene and protein names (Jiao et al., 2006) and for named entity recognition and part-of-speech tagging (Mann and McCallum, 2007) and transductive support vector machines for mapping sentences onto a formal meaning representation (Kate and Mooney, 2007) and performing dependency parsing (Wang et al., 2008). We refer the interested reader to (Abney, 2007) for more references on semi-supervised learning for natural language processing. Although most of this research shows that semi-supervised learning for natural language processing improves results, some researchers have also pointed to cases where adding unlabeled examples deteriorated performance. Nigam et al. (2000) find that a probabilistic model that does not capture dependencies between features or that does not approximate the correct number of clusters in the data can result in performance degradation for semi-supervised methods. Charniak (1997) and Pierce and Cardie (2001) find that respectively self-training for sentence parsing and co-training for noun phrase bracketing can lead to increased errors in the classification, most likely caused by incorporating incorrectly labeled examples in the training phase.

In recent years semi-supervised learning methods have been applied to semantic role labeling. He and Gildea (2006) use a self-learning scheme where a maximum entropy classifier is trained using a small set of labeled examples. This classifier is then used to create semantic role labellings for a large set of unlabeled sentences, of which the most confident are added to the labeled training set. This process is repeated for a number of iterations. Contrary to expectations this did not improve the performance of the classifier. Also a co-training scheme where two classifiers were trained independently and used iteratively to label unlabeled sentences failed to improve performance.

Swier and Stevenson (2004) report on a successful self-learning method to learn VerbNet semantic roles (Kipper et al., 2000) where the probabilistic classifier is augmented with a set of linguistic restrictions to guide the assignment of semantic roles to sentence constituents, improving the performance of a baseline method from a 63.7% accuracy to a 87.2% accuracy on a labeled section of the British National Corpus for 54 target verbs. The method was only tested on a small subset of all verbs in VerbNet and to the best of our knowledge has not been reproduced on the much larger PropBank dataset. Furthermore we can see linguistic constraints as an alternative to labeled examples, adding extra human knowledge to the classifier.

Finally we cite Fürstenau and Lapata (2009) who compute a syntactic and semantic distance between sentence constituents to automatically expand a small training set with the most similar sentences in a large set of unlabeled examples. The similarity metric combines a syntactic distance (using the dependency tree of the sentence) and a semantic distance metric (using the Brown clustering algorithm (Brown et al., 1992)). A supervised classifier is then trained on this expanded set. This method was successfully tested on the FrameNet corpus, where the largest relative improvements were achieved for small initial sets of labeled examples. We will discuss this method more at length in chapter 7, where we will compare it with a method that combines structures learned by an unsupervised model.

5.6 Conclusions of this chapter

In this chapter we have introduced semi-supervised learning as a solution to the underspecification problem and discussed how semi-supervised learning is based on the semi-supervised smoothness assumption. We have then extended the Bayesian networks with hidden variables for the unlabeled examples and shown how the parameters can be estimated with Markov Chain Monte Carlo sampling, which we applied on the generative and discriminative SRL models. The parameters of the generative SRL model were estimated with Gibbs sampling and the parameters of the discriminative model with Metropolis-Hastings sampling. We observed how the performance of these semi-supervised classifiers deteriorated when using more unlabeled data, which is caused by a violation of the correct model assumption. A multiple-mixtures model was then proposed as an improvement to the generative model, allowing more degrees of freedom when modeling natural language. We saw how the performance of this model was more robust when adding more unlabeled data.

From these experiments we regrettably have to conclude that Bayesian networks with hidden variables are not a suitable paradigm for semi-supervised learning of information extraction methods. This is in line with results indicating that semi-supervised learning, although potentially usefull for simple information extraction tasks (e.g. part-of-speech tagging (Cutting et al., 1992) and named entity recognition (Collins and Singer, 1999)), does not help for more complex tasks (e.g. noun phrase chunking (Pierce and Cardie, 2001) and syntactic sentence parsing (Charniak, 1997)). In the next chapter we turn to a different approach to weakly supervised learning: we first train an unsupervised model on unlabeled data, and use, in a second step, the statistics learned by this model in supervised classifier trained on annotated data. We will see how this approach depends only slightly on the correct model assumption, and does lead to improved results when used with large amounts of unlabeled data.

Chapter 6

The Latent Words Language Model

“*When ideas fail, words come in very handy*”
Johann Wolfgang von Goethe

In this section we discuss a novel unsupervised model of natural language, the latent words language model. This model learns syntactically and semantically similar words from a large corpus of unlabeled texts to improve the predictive quality of an n-gram language model on unseen texts. We will start by describing state-of-the-art n-gram language models and the problems encountered with these models (section 6.1) and introduce the LWLM as a possible solution to these problems (section 6.2). We evaluate the predictive quality of this model on unseen texts in section 6.3 and the learned word similarities in section 6.4. We discuss related work in section 6.5 and conclude this chapter in section 6.6.

Although this chapter might seem to be deviating from the topic of this thesis, we ask the reader for some patience, since we will show at the end of this chapter and in the next chapter that the structures learned in this language model are very useful to improve the performance of information extraction methods.

6.1 N-gram language models

Language models are models that assign a probability to every sequence of words $\mathbf{w} = [w_1 ... w_N]$, which reflects the probability that this sequence will be generated by a human user of natural language. Sentences that are likely to be uttered should thus be assigned a higher probability and sentences that are unlikely to be uttered a lower probability. These models have been used in a wide range of applications, such as speech recognition (Jelinek et al., 1975), machine translation (Brown et al., 1990), spelling correction (Kemighan et al., 1990) and handwriting recognition (Srihari and Baltus, 1992). In this section we introduce the most successful class of language models, n-gram language models.

6.1.1 Introduction

Although any probabilistic method can be used for language modeling, the most successful language models are n-gram models. These models estimate the probability $p(\mathbf{w})$ of the sequence of words $\mathbf{w} = [w_1 ... w_N]$ as

$$p(\mathbf{w}) = \prod_{i=1}^{N} P(w_i|\mathbf{w}_{i-n+1}^{i-1})$$

where $\mathbf{w}_{i-n+1}^{i-1} = [w_{i-n+1} ... w_{i-1}]$ is the sequence of $n-1$ words that occur before w_i, i.e. the probability of w_i is computed using only the $n-1$ previous words. The value of n is usually set to a small number (e.g. 3). These models are trained on a large unlabeled corpus $\mathbf{w}_{train} = [w_1 ... w_{N_t}]$. Let us first consider the maximum likelihood estimate of the probability $P(w_i|\mathbf{w}_{i-n+1}^{i-1})$, given by

$$P_{ML}(w_i|\mathbf{w}_{i-n+1}^{i-1}) = \frac{c(\mathbf{w}_{i-n+1}^{i})}{c(\mathbf{w}_{i-n+1}^{i-1})}$$

where $c(\mathbf{w}_{i-n+1}^{i})$ is the number of times the sequence of words $\mathbf{w}_{i-n+1}^{i}$ occurs in $\mathbf{w}_{train}$ and $c(\mathbf{w}_{i-n+1}^{i-1})$ is the number of times the sequence $\mathbf{w}_{i-n+1}^{i-1}$ occurs in $\mathbf{w}_{train}$. This is a proper probability distribution since $\sum_{w_i} c(\mathbf{w}_{i-n+1}^{i}) = c(\mathbf{w}_{i-n+1}^{i-1})$, but it will lead to an ill-defined model, since it assigns zero probability to many sequences. The reason for this is that potentially $|V|^n$ n-grams can occur in a given corpus, where $|V|$ is the size of the vocabulary (usually between 10^4 and 10^6 words). An unobserved test corpus is thus likely to contain many sequences that have never been observed. Figure 6.1 shows that the probability of an n-gram in a test corpus being observed in the training corpus becomes exponentially smaller with increasing n.

Methods to overcome this problem generally combine the probability of $\mathbf{w}_{i-n}^{i}$ with lower order probabilities, i.e. the probability of observing $\mathbf{w}_{i-n+1}^{i}$, the probability

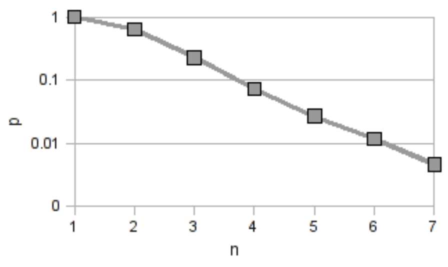


Figure 6.1: Probability p that an n-gram of length n in the test section has been observed in the training section, both from the Reuters corpus (section 6.3).

of observing $\mathbf{w}_{i-n+2}^{i}$, ... and the probability of observing w_i. For an extensive overview of these methods we refer to Chen and Goodman (1996) and Goodman (2001). In the following sections we review some methods relevant to the research at hand: a simple interpolation model (section 6.1.2), the state-of-the-art Kneser-Ney smoothing method (section 6.1.3) and the novel relative discount Kneser-Ney smoothing method (section 6.1.4).

6.1.2 Interpolation

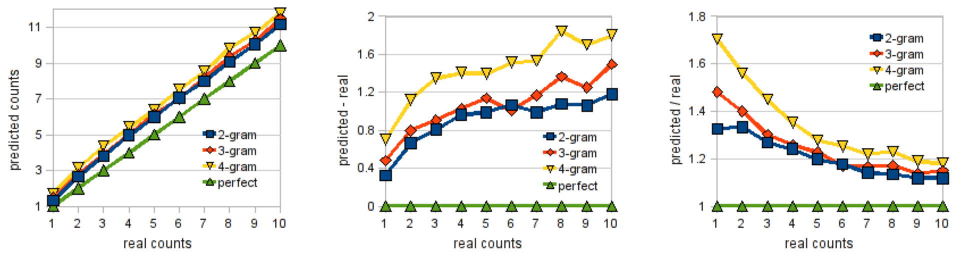
A first method that is discussed is a simple interpolation model. The probability of the word w_i given the previous words $\mathbf{w}_{i-n+1}^{i-1}$ is computed by the interpolation model as

$$P_{INT}(w_i|\mathbf{w}_{i-n+1}^{i-1}) = \lambda_n \frac{c(\mathbf{w}_{i-n+1}^{i})}{c(\mathbf{w}_{i-n+1}^{i-1})} + (1-\lambda_n)P_{INT}(w_i|\mathbf{w}_{i-n+2}^{i-1}) \tag{6.1}$$

where λ_n is a smoothing factor $0 \leq \lambda_n \leq 1$. This is a recursive definition, in which the last term is the unigram probability $P_{INT}(w_i) = \frac{c(w_i)}{N_t}$, i.e. the relative frequency of the word w_i in the training corpus. This method combines specific, but sparse and thus possibly unreliable, higher order n-grams with less specific, but more reliable, lower order n-grams. The interpolation factors λ_n are constants selected to optimize the predictive quality of this model (as measured by the likelihood of a held-out corpus, see section 6.3).

6.1.3 Kneser-Ney smoothing

One problem with simple interpolation is that not all frequency counts should be considered equally reliable. Consider the sequence of words “Garry Tucker said” in the $5M$ words Reuters corpus (section 6.3). This phrase is only observed this one



(a) predicted counts (b) absolute difference (c) relative difference

Figure 6.2: Comparison of the real frequency in a $50M$ Reuters corpus versus the expected frequency based on a $5M$ corpus.

time in the entire corpus. The maximum likelihood estimate (i.e. $\frac{1}{10^5}$) is however most likely a serious overestimation, since most likely we will not observe this phrase 10 times in a corpus of $50M$ words. Figure 6.2a shows the relation between the expected count (based on a $5M$ corpus) and the observed count (based on a disjunct $50M$ corpus in the same domain). We see that on average, the expected count is larger than the observed count. This difference can be quantified in absolute terms (i.e. the value obtained by subtracting the average predicted counts from the average measured counts, figure 6.2b) and in relative terms (i.e. the value obtained by dividing the average predicted counts by the average measured counts, figure 6.2c). We see how both differences changes with the frequency: low frequencies have a small absolute difference but a large relative difference, while larger frequencies have a bigger absolute difference but smaller relative difference.

These observations suggest that a more accurate distribution can be obtained by discounting $c(\mathbf{w}_{i-n+1}^{i})$ with some factor. This is incorporated in the absolute discounted Kneser-Ney smoothing method, proposed by Ney et al. (1994) and adapted by Chen and Goodman (1996)

$$P_{akn}(w_i|\mathbf{w}_{i-n+1}^{i-1}) = \frac{c(\mathbf{w}_{i-n+1}^{i}) - d_n(c(\mathbf{w}_{i-n+1}^{i}))}{c(\mathbf{w}_{i-n+1}^{i-1})} + \delta(\mathbf{w}_{i-n+1}^{i-1}) P_{akn}(w_i|\mathbf{w}_{i-n+2}^{i-1}) \tag{6.2}$$

where $d_n(c(\mathbf{w}_{i-n+1}^{i-1}))$ is the discount factor for count $c(\mathbf{w}_{i-n+1}^{i-1})$ and $\delta(\mathbf{w}_{i-n+1}^{i-1})$ is an interpolation factor that combines the n-gram distribution with the lower order $n-1$-gram distribution and is defined by

$$\delta(\mathbf{w}_{i-n+1}^{i-1}) = 1 - \sum_{w_i} \frac{c(\mathbf{w}_{i-n+1}^{i}) - d_n(c(\mathbf{w}_{i-n+1}^{i}))}{c(\mathbf{w}_{i-n+1}^{i-1})} \tag{6.3}$$

A common choice (Chen and Goodman, 1996) for the discount factor is

$$d_n(c(\mathbf{w}_{i-n}^i)) = \begin{cases} 0 & if\, c(\mathbf{w}_{i-n}^i) = 0 \\ d_{n1} & if\, c(\mathbf{w}_{i-n}^i) = 1 \\ d_{n2} & if\, c(\mathbf{w}_{i-n}^i) = 2 \\ d_{n3+} & otherwise \end{cases}$$

where d_{n1}, d_{n2} and d_{n3+} are constants optimized on a held-out corpus. Unique to Kneser-Ney smoothing is that the probabilities of bigrams and unigrams are computed in a different manner. The probability distribution of bigrams is given by

$$P_{akn}(w_i|w_{i-1}) = \frac{\pi(w_{i-1}w_i) - d_2(\pi(w_{i-1}w_i))}{\sum_{w_j} \pi(w_{i-1}w_j)} + \delta(w_{i-1})P_{akn}(w_i) \tag{6.4}$$

where $\pi(w_{i-1}w_i) = |\{v|c(vw_{i-1}w_i) > 0\}|$ is the number of different words v such that the sequence $vw_{i-1}w_i$ occurs at least once in the training set. Similarly the unigram probability is computed as

$$P(w_i) = \frac{\pi(w_i)}{\sum_{w_j} \pi(w_j)} \tag{6.5}$$

where $\pi(w_i) = |\{v|c(vw_i) > 0\}|$ is the number of different words v such that the sequence vw_i at least once. These special computations of the bigram and unigram distributions are motivated by the observation that some words (e.g. "Francisco") occur frequently in a corpus, but occur only in very specific contexts (e.g. "San Francisco"), and that counting the number of unique contexts such a word occurs in gives a better estimate of the true probability of observing this word in a new context.

This smoothing method combines a number of ideas: (1) the maximum likelihood estimate is closer to the true distribution if the raw counts are discounted with an absolute factor, (2) lower order counts are better estimated with formula's 6.4 and 6.5 to match the marginals of the higher-order distributions to the marginals of the training data, and (3) depending on the structure of the higher order distribution, more or less weight should be given to this distribution. The motivation for (1) was given previously and for the motivation for (2) we refer to (Chen and Goodman, 1996). Property (3) however is an interesting quality that we will discuss a bit more in depth.

The interpolation factor $\delta(\mathbf{w}_{i-n+1}^{i-1})$, given by equation 6.3, is not a static factor, but a dynamic factor that depends on the shape of the higher order distribution.

Assume for example the following definition for the discount factor

$$d_n(c(\mathbf{w}_{i-n+1}^{i})) = \begin{cases} 0 & if\, c(\mathbf{w}_{i-n+1}^{i}) = 0 \\ 0.9 & if\, c(\mathbf{w}_{i-n+1}^{i}) = 1 \\ 1.5 & if\, c(\mathbf{w}_{i-n+1}^{i}) = 2 \\ 2.1 & otherwise \end{cases}$$

Take that we observe the 3-gram "Garry Tucker said", and want to compute the probability distribution of the next word. "Garry Tucker said" occurs only once in the training corpus and the interpolation factor $\theta(\text{"Garry Tucker said"})$ is thus $1 - \frac{1-0.9}{1} = 0.9$. A high weight is thus given to the lower order probability distributions, since the higher order distribution was sparse and probably not very reliable.

Now take that we observe the 3-gram "the first quarter", which occurs 5493 times in the corpus, and occurs frequently with the same 4-grams e.g. "the first quarter of" (2643 times), "the first quarter , " (708 times) and "the first quarter and" (187 times). For this training corpus, we find that $\delta(\text{"the first quarter"}) = 0.05$, giving a high weight tot the higher order distribution since it was observed frequently and is probably reliable.

Kneser-Ney smoothing is not the only smoothing method to incorporate this dynamic interpolation (e.g. Jelinek and Mercer (1980) and Bell et al. (1990)), but in our opinion it is the only method to combine this in an effective way with the discount of individual sequences. In an extensive comparison of a large number of smoothing techniques, Chen and Goodman (1996) found that interpolated Kneser-Ney smoothing consistently outperforms all other state-of-the-art smoothing methods.

6.1.4 Relative discounted Kneser-Ney smoothing

One disadvantage of interpolated Kneser-Ney smoothing is that it is only defined for discrete counts. In the following sections we will use it in the EM-algorithm, where we need to discount non-discrete, probabilistic counts. It is also unclear how to count the number of unique contexts of a particular word in this setting. For these reasons we propose a novel modification of the Kneser-Ney smoothing method, termed relative discounted Kneser-Ney smoothing (RDKN). RDKN uses a relative discount factor $d_n(c(\mathbf{w}_{i-n+1}^{i}))$ between 0 and 1. The smoothing method $P_{rkn}(w_i|\mathbf{w}_{i-n+1}^{i-1})$ is then defined by

$$P_{rkn}(w_i|\mathbf{w}_{i-n+1}^{i-1}) = \frac{c(\mathbf{w}_{i-n+1}^{i}) \times d_n(c(\mathbf{w}_{i-n+1}^{i}))}{c(\mathbf{w}_{i-n+1}^{i-1})} + \delta(\mathbf{w}_{i-n+1}^{i-1}) P_{rkn}(w_i|\mathbf{w}_{i-n+2}^{i-1}) \tag{6.6}$$

where the recursion ends with the unigram probability $P_{rkn}(w_i) = \frac{c(w_i)}{N}$. As before the interpolation factor $\delta(\mathbf{w}_{i-n+1}^{i-1})$ is defined as

$$\delta(\mathbf{w}_{i-n+1}^{i-1}) = 1 - \sum_{w_i} \frac{c(\mathbf{w}_{i-n+1}^{i}) \times d_n(c(\mathbf{w}_{i-n+1}^{i}))}{c(\mathbf{w}_{i-n+1}^{i-1})} \tag{6.7}$$

We define $d_n(c(\mathbf{w}_{i-n+1}^{i}))$ by dividing the space of counts $c(\mathbf{w}_{i-n+1}^{i})$ into S equally populated intervals with borders c_1, ... , c_{S-1}. Every interval is assigned a fixed discount factor d_s, defining the function $d_n(c(\mathbf{w}_{i-n+1}^{i}))$ as

$$d_n(c(\mathbf{w}_{i-n+1}^{i})) = \begin{cases} d_{n1} & if\, c(\mathbf{w}_{i-n+1}^{i}) \leq c_1 \\ d_{n2} & if\, c_1 < c(\mathbf{w}_{i-n+1}^{i}) \leq c_2 \\ ... & ... \\ d_{nS} & if\, c_S < c(\mathbf{w}_{i-n+1}^{i}) \end{cases}$$

The values $0 \leq d_{ni} \leq 1$ are optimized on a held-out corpus.

This method is different to interpolated Kneser-Ney smoothing in two aspects. The first difference is that we use a relative discount factor and not an absolute discount factor. The second difference is that we do not use special counts for bigrams and unigram distributions, but use the relative discounted counts as given by equation 6.6 for the bigram distribution and use the maximum likelihood estimate for the unigram distribution.

This formulation can be used conveniently with soft counts, and in fact we see in section 6.3 that it also outperforms Kneser-Ney smoothing when used with discrete counts.

6.2 The latent words language model

As discussed in the previous section, the performance of language models is limited by the sparse nature of n-grams. Although smoothing methods partially alleviate this problem, they do not fully solve it. A major weakness of these models is that they treat every word in the text as a unique symbol, independent of all other symbols. This ignores the fact that many words are synonyms or have related meanings, and that natural language typically uses constructions where certain classes of words (e.g. part-of-speech classes) always occur on the same position (e.g. "determiner noun verb"). Let us assume for example that we observe the sequence "let's meet on Tuesday" in the training corpus. If the model would know that "Tuesday" is similar to "Monday", "Wednesday" etc., we could predict that the sequences "let's meet on Monday", "let's meet on Wednesday", etc. can also be observed in the test corpus.

In this section we build a model that aims at exactly this goal: learning words that are synonyms or that have related meanings, and use these in an improved model for the prediction of sequences in the test corpus. We first describe the model in section 6.2.1. Although the definition of the model is simple, standard algorithms can not be employed due to their large time complexity. We discuss novel algorithms for inference (section 6.2.2), training (section 6.2.3) and predicting the probability of unseen texts (section 6.2.4). Finally we discuss some additional techniques used in the implementation in section 6.2.5.

6.2.1 Description of the model

The latent words language model (LWLM) introduces for a text $\mathbf{w} = [w_1...w_N]$ of length N for every observed word w_i at position i a hidden (or latent) word h_i with an unknown value from the vocabulary V. This model is a generative model for natural language that, for a given vocabulary V, length N, counts C and smoothing parameters γ, generates a sequence of hidden symbols $\mathbf{h} = [h_1...h_N]$ and a sequence of observed words $\mathbf{w} = [w_1...w_N]$. The generative process is defined as follows:

```
For $i$ from 1 to $N$ do
        Sample a hidden word $h_i$ from the distribution $P(h_i|\mathbf{h}_{i-n}^{i-1}, C, \gamma)$
        Sample an observed word $w_i$ from the distribution $P(w_i|h_i, C, \gamma)$
```

Here we implicitly understand that for the first words in the sequence, i.e. $i < n$, we only use the available context, e.g. if $i = 1$ then $P(h_i|\mathbf{h}_{i-n}^{i-1}, C, \gamma) = P(h_i|C, \gamma)$.

This model contains two probability distributions. The first distribution, $P(h_i|\mathbf{h}_{i-n}^{i-1}, C, \gamma)$, models the dependency between the current hidden word and the previous hidden words, and is modeled as a categorical distribution, where the raw counts are smoothed with relative discounted Kneser-Ney smoothing (RDKN).

The second distribution $P(w_i|h_i, C, \gamma)$ models the dependency of the observed word on the hidden word and is also a categorical distribution, smoothed with a

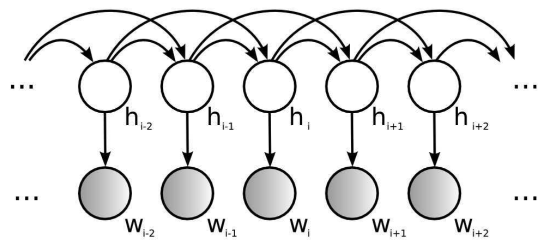


Figure 6.3: BN of the latent words language model. The words w_i (gray nodes) are observed and the hidden words h (white nodes) are hidden variables.

variant of RDKN

$$P(w_i|h_i, C, \gamma) = \frac{c(w_i, h_i) \times d(c(w_i, h_i))}{c(h_i)} + \delta(h_i)\frac{c(w_i)}{N_t} \tag{6.8}$$

We will use the term “smoothing parameters”, with symbol γ, to denote the collection of discount factors used in these two smoothing methods.

This model can also be expressed as a Bayesian network, shown in figure 6.3. From this figure we see that the structure of the model is equivalent to the structure of a hidden Markov model (HMM) (Baum et al., 1970; Baker, 1975). However, the two models have important differences: where the latent variables in a **2** are selected from a small set of categories, we model the latent variables as unseen words that can be selected from the entire vocabulary. Furthermore we use a novel smoothing method for the context model (i.e. RDKN), and algorithms for HMM's typically assume that the hidden variables are only dependent on the previous variable (i.e. $n = 2$), where we will use much longer dependencies (i.e. $n \geq 5$), which has important consequences for the algorithms employed for training and inference.

One interpretation of this model states that a person who wishes to express a certain message, can choose a large number of ways of expressing this message. However, when the message is uttered (or written down), this person has to chose one specific sequence of words, although many words in this sequence could be replaced with a synonym or related word while keeping the meaning of the message intact. The hidden word h_i can be seen to represent these possible alternative words at a certain position i in this sequence. The collection of possible alternatives is modeled as a probability distribution $P(h_i|\mathbf{w})$ over all words in the vocabulary.

This interpretation is of course very loosely defined, and probably also a bit over-ambitious. We will however see that by training this model on a large collection of unlabeled texts, we can learn word similarities with a high accuracy. These word similarities can then be used successfully to improve language models and information extraction methods.

We develop three methods for this model: *inference*, where we estimate the expected value of the hidden variables given the observed words, *training*, where we estimate the parameters of the model given a large training text, and *density estimation*, where we predict the probability of unseen texts.

6.2.2 Inference

Given a sequence of words $\mathbf{w}_{test} = [w_1 ... w_{N_u}]$ and the parameters C and γ, we want to find the probability distribution $P(h_i|\mathbf{w}_{test}, C, \gamma)$ of the hidden word h_i on position i.

The forward-backward algorithm Traditionally in HMM's this probability is computed with the forward-backward (i.e. Baum-Welch) algorithm, which computes this probability as

$$P(h_i|\mathbf{w}_{test}, C, \gamma) = \sum_{\mathbf{h}_{i-n+2}^{i-1}} \frac{\alpha(\mathbf{h}_{i-n+2}^{i})\beta(\mathbf{h}_{i-n+2}^{i})}{P(\mathbf{w}_{test}|C, \gamma)} \tag{6.9}$$

where $\alpha(\mathbf{h}_{i-n+2}^{i}) = P(\mathbf{w}_1^i, \mathbf{h}_{i-n+2}^{i}, C, \gamma)$ is the joint probability of observing $\mathbf{w}_1^i$ together with the sequence $\mathbf{h}_{i-n+2}^{i}$, and $\beta(\mathbf{h}_{i-n+2}^{i}) = P(\mathbf{w}_{i+1}^{N_u}|\mathbf{h}_{i-n+2}^{i}, C, \gamma)$ represents the conditional probability of observing $\mathbf{w}_{i+1}^{N_u}$ given the sequence $\mathbf{h}_{i-n+2}^{i}$. Both values are defined recursively as

$$\alpha(\mathbf{h}_{i-n+2}^{i}) = P(w_i|h_i, C, \gamma) \sum_{h_{i-n+1}} \alpha(\mathbf{h}_{i-n+1}^{i-1}) P(h_i|\mathbf{h}_{i-n+1}^{i-1}, C, \gamma) \tag{6.10}$$

and

$$\beta(\mathbf{h}_{i-n+2}^{i}) = \sum_{h_{i+1}} \beta(\mathbf{h}_{i-n+3}^{i+1}) P(w_{i+1}|h_{i+1}, C, \gamma) P(h_{i+1}|\mathbf{h}_{i-n+2}^{i} C, \gamma) \tag{6.11}$$

The values $\alpha(\mathbf{h}_{i-n+2}^{i})$ can been seen as messages that travel from the start of the sequence to the end, while the values $\beta(\mathbf{h}_{i-n+2}^{i})$ can be seen as messages that travel from the end of the sequence to the start. As the forward and backward messages travel through the chain, at every position in the sequence we need to store the values of $\alpha(\mathbf{h}_{i-n+2}^{i})$ and $\beta(\mathbf{h}_{i-n+2}^{i})$ for every possible sequence $\mathbf{h}_{i-n+2}^{i}$, requiring the storage of $2 \times |H|^{n-1}$ values, where $|H|$ is the number of hidden states. To compute the next $\alpha(\mathbf{h}_{i-n+3}^{i+1})$ (or $\beta(\mathbf{h}_{i-n+1}^{i-1})$), we need to multiply and sum these values for every possible hidden state in the next position, requiring a computational cost of $O(|H|^{n})$. This needs to be performed for every position, resulting in a total complexity of $O(|H|^{n} \times N_u)$. For more information on the forward-backward algorithm we refer to (Bishop, 2006).

The computational cost of the forward-backward algorithm for standard HMM's is low because most implementations of hidden Markov models use only the previous context, i.e. $n = 2$, and use a small number of hidden states, i.e. $10^0 < |H| < 10^2$. In our model the number of the hidden variables is the size of the vocabulary, which is $10^4 < |V| < 10^6$ for any reasonable sized corpus, and we want to use a much larger context, e.g. $n = 5$. It is thus clear that using the traditional forward-backward algorithm is not an option here.

Forward-forward beam search We develop an approximate version of the forward-backward algorithm with lower time complexity, termed the *forward-forward beam search*. We introduce a function $trim(\alpha(\mathbf{h}_{i-n+2}^{i}))$ defined by

$$trim(\alpha(\mathbf{h}_{i-n+2}^{i})) = \begin{cases} \alpha(\mathbf{h}_{i-n+2}^{i}) & if\, rank(\alpha(\mathbf{h}_{i-n+2}^{i})) \geq b \\ 0 & otherwise \end{cases}$$

where the rank $rank(\alpha(\mathbf{h}_{i-n+2}^{i}))$ is found by sorting all possible sequences $\mathbf{h}_{i-n+2}^{i}$ in descending order according to the value of $\alpha(\mathbf{h}_{i-n+2}^{i})$. The trim function thus removes for a certain position all values that are not among the b most likely values. We then define the approximate $\alpha'(\mathbf{h}_{i-n+2}^{i})$[1]

$$\alpha'(\mathbf{h}_{i-n+2}^{i}) = \sigma_{i-n+1} P(w_i|h_i) \sum_{h_{i-n+1}} trim(\alpha'(\mathbf{h}_{i-n+1}^{i-1})) P(h_i|\mathbf{h}_{i-n+1}^{i-1}) \qquad (6.12)$$

this value is an approximation of $\alpha(\mathbf{h}_{i-n+2}^{i})$ that takes into account only the b most likely values of $\alpha'(\mathbf{h}_{i-n+1}^{i-1})$. This value will be closer to $\alpha(\mathbf{h}_{i-n+2}^{i})$ with increasing b, and will be equal if $b \geq |H|^{n-1}$. The factor σ_{i-n+1} is the inverse of the probability mass not discarded on position $i - n + 2$, given by

$$\sigma_{i-n+1} = \frac{\sum_{\mathbf{h}_{i-n+2}^{i}} \alpha'(\mathbf{h}_{i-n+1}^{i-1})}{\sum_{\mathbf{h}_{i-n+2}^{i}} trim(\alpha'(\mathbf{h}_{i-n+1}^{i-1}))}$$

This factor makes sure that the total probability mass in the network remains equal to 1.0. On every position we need to perform the summation in equation

[1]For brevity we omit the parameters C and γ from these and the following equations.

Algorithm 1 Forward beam search for text $\mathbf{w}_{N_t}$.

```
Require: $V$, $C$, $\gamma$, $\mathbf{w}_{train}$, $d$, $n$, $b$
Ensure: $A_i = \alpha'(\mathbf{h}^i_{i-n+2})$ for $1 \leq i \leq N_t$
 1: $A_0 \Leftarrow \{([\,], 1.0)\}$
 2: for $i = 1$ to $N_t$ do
 3:   $A_i \Leftarrow \{\}$
 4:   for all $(\mathbf{h}^{i-1}_{i-n+1}, \alpha)$ in $A_{i-1}$ do
 5:     for all $h_i$ in V do
 6:       $\alpha^* \Leftarrow \alpha \times P(h_i|\mathbf{h}^{i-i}_{i-n+1}, C, \gamma) \times P(w_i|h_i, C, \gamma)$
 7:       $\mathbf{h}^i_{i-n+2} \Leftarrow [\mathbf{h}^{i-1}_{i-n+2} h_i]$
 8:       $A_i \Leftarrow A_i \cup \{(\mathbf{h}^i_{i-n+2}, \alpha^*)\}$
 9:     end for
10:   end for
11:   $A_i \Leftarrow$ sumSame($A_i$)
12:   $A_i \Leftarrow$ trim($A_i$,$b$)
13: end for
```

6.12 with a time complexity of $O(b|V|)$, and we need to sort these values to find the b most likely values in the next iteration, with complexity $O(b|V|\log(b|V|))$.

The pseudo-code for the forward beam search is shown in algorithm 1. When this algorithm has been executed for a given text $\mathbf{w}_{train}$, the collection A_i contains all $\alpha'(\mathbf{h}^i_{i-n+2})$ for position i. The functions $sumSame(A_i)$ and $trim(A_i)$ are not shown in the algorithm and perform the following operations on A_i : $sumSame(A_i)$ sums the probabilities of all structures $(\mathbf{h}^j_{j-n+2}, \alpha)$ in A_i that have the same value for $\mathbf{h}^j_{j-n+2}$. The function $trim(A_i, b)$ is the implementation of the $trim(\alpha'(\mathbf{h}^{i-1}_{i-n+1}))$ operator introduced earlier, i.e. it sorts all the values according to decreasing α, selects the b most likely and rescales the remaining values with σ_{i-n+1} to make sure that the total probability mass remains 1.0.

We could define a similar trimmed version of $\beta(\mathbf{h}^i_{i-n+2})$, but here we face an additional difficulty. We use relative discounted Kneser-Ney smoothing to compute $P(h_{i+1}|\mathbf{h}^i_{i-n+2}, C, \gamma)$, which requires the computation of $\delta(\mathbf{h}^i_{i-n+2})$. This value depends on all possible values for h_{i-n+2}, and limiting the computation to only the b most likely values of h_{i-n+2} would result in a degenerate probability distribution.

For this reason we introduce a new forward probability $\gamma(\mathbf{h}^i_{i-n+2}, h_j)$ which is the joint probability of observing the words $\mathbf{w}^i_1$, the sequence $\mathbf{h}^i_{i-n+2}$ and the hidden variable h_j. This value is defined for $i \geq j$ and is given by

$$\gamma(\mathbf{h}^i_{i-n+2}, h_j) = \begin{cases} P(w_i|h_i)\sum_{h_{i-n+1}} \alpha(\mathbf{h}^{i-1}_{i-n+1})P(h_i|\mathbf{h}^{i-1}_{i-n+1}) & if\, i = j \\ P(w_i|h_i)\sum_{h_{i-n+1}} \gamma(\mathbf{h}^{i-1}_{i-n+1}, h_j)P(h_i|\mathbf{h}^{i-1}_{i-n+1}) & if\, i > j \end{cases}$$

This value is similar to $\alpha(\mathbf{h}_{i-n+2}^{i})$, but also includes the probability of generating h_j. We can interpret $\gamma(\mathbf{h}_{i-n+2}^{i}, h_j)$ as a series of messages, where for every different value of h_j, a series of messages $\gamma(\mathbf{h}_{i-n+2}^{i}, h_j)$ is passed from position j to the end of the sequence. A trimmed version of this variable is defined as

$$\gamma'(\mathbf{h}_{i-n+2}^{i}, h_j) = \begin{cases} \sigma_{i-n+1} P(w_i|h_i) \sum_{h_{i-n+1}} trim(\alpha'(\mathbf{h}_{i-n+1}^{i-1})) P(h_i|\mathbf{h}_{i-n+1}^{i-1}) & if\, i = j \\ \sigma_{i-n+1} P(w_i|h_i) \sum_{h_{i-n+1}} trim(\gamma'(\mathbf{h}_{i-n+1}^{i-1}, h_j)) P(h_i|\mathbf{h}_{i-n+1}^{i-1}) & if\, i > j \end{cases}$$

The sum in this equation has time complexity $O(b|V|)$ and sorting the values has complexity $O(b|V|\log(b|V|))$. Doing so for every position in the sequence results in a time complexity of $O(N_u \times [b|V|\log(b|V|)])$.

After passing all messages to the end of the sequence, we have a collection of messages $\gamma'(\mathbf{h}_{N_u-n+2}^{N_u}, h_j)$. To compute the probability of h_j given $\mathbf{w}_{test}$ we sum over all possible values of $\mathbf{h}_{N_u-n+2}^{N_u}$, i.e.

$$P(h_j|\mathbf{w}_{test}, C^\tau) = \frac{\sum_{\mathbf{h}_{N_u-n+2}^{N_u}} \gamma(\mathbf{h}_{N_u-n+2}^{N_u}, h_i)}{P(\mathbf{w}_{test}|C^\tau)} \simeq \frac{\sum_{\mathbf{h}_{N_u-n+2}^{N_u}} \gamma'(\mathbf{h}_{N_u-n+2}^{N_u}, h_i)}{P(\mathbf{w}_{test}|C^\tau)}$$

Summarizing, we first pass a series of forward messages $\alpha(\mathbf{h}_{i-n+1}^{i-1})$ from the start of the sequence to the end. We then pass for every position j in the sequence, a series of $\gamma(\mathbf{h}_{i-n+1}^{i-1}, h_j)$ messages from that position to the end of the sequence. One important disadvantage of this approach is that in the original forward-backward algorithm only a single pass of forward messages and a single pass from backward messages is required to compute the probability distribution for every h_i. In our formulation however we first perform a single pass of forward messages, and then perform, for every position i, a pass of forward messages $\gamma(\mathbf{h}_{j-n+2}^{j}, h_i)$ from position i to the end of the sequence. This greatly increases the complexity of the algorithm. Therefore we make an additional assumption: we assume that the probability distribution of h_i given $\mathbf{w}_1^{N_t}$ is approximately equal to the probability of h_i given the words $\mathbf{w}_1^{i+\delta}$ for a certain distance δ.

$$P(h_i|\mathbf{w}_1^{N_t}, C^\tau) \simeq P(h_i|\mathbf{w}_1^{i+\delta}, C^\tau)$$

This seems plausible since the words that occur in the sequence far away from the hidden word will have little influence on this word.

The total time complexity of the forward-forward beam search is $O(N_u \times (1+d) \times (b|V| + b|V|\log(b|V|))$, which will be, even for fairly high values for b and d, much lower than the original time complexity of the forward-backward algorithm.

The pseudo-code for the *forward-forward beam search* is shown in algorithm 2. The $\gamma'(\mathbf{h}_{j-n+1}^{j-1}, h_i)$ values are stored in a data structure $\Gamma_j = \{(\mathbf{h}_{j-n+1}^{j-1}, h_i, \gamma)\}$

Algorithm 2 Computation of expected values of $h_1, ..., h_{N_t}$ given $\mathbf{w}_{N_t}$

```
Require: $V$, $C$,$\gamma$, $\mathbf{w}_{train}$, $\delta$, $b$, $A_1, ..., A_{N_t}$
Ensure: $H_j \simeq P(h_j|\mathbf{w}_{train}, C, \gamma)$ for $1 \leq j \leq N_t$
 1: for $j = 1$ to $N_t$ do
 2:   $\Gamma_j \Leftarrow \{\}$
 3:   for all $(\mathbf{h}^{j}_{j-n+2}, \alpha)$ in $A_j$ do
 4:     $\Gamma_j \Leftarrow \Gamma_j \cup \{(\mathbf{h}^{j}_{j-n+2}, h_j, \alpha)\}$
 5:   end for
 6:   for $i = j + 1$ to $j + \delta$ do
 7:     $\Gamma'_i \Leftarrow \{\}$
 8:     for all $(\mathbf{h}^{i-1}_{i-n+1}, h_j, \gamma)$ in $\Gamma_{i-1}$ do
 9:       for all $h_i$ in V do
10:         $\gamma^* \Leftarrow \gamma \times P(h_i|\mathbf{h}^{i-1}_{i-n+1}, C, \gamma) \times P(w_i|h_i, C, \gamma)$
11:         $\mathbf{h}^{i}_{i-n+2} \Leftarrow [\mathbf{h}^{i-1}_{i-n+2} h_i]$
12:         $\Gamma'_i \Leftarrow \Gamma'_i \cup \{(\mathbf{h}^{i}_{i-n+2}, h_j, \gamma^*)\}$
13:       end for
14:     end for
15:     $\Gamma'_i \Leftarrow$ sumSame($\Gamma'_i$)
16:     $\Gamma_i \Leftarrow$ trim($\Gamma'_i$,$b$)
17:   end for
18:   $H_j \Leftarrow \{\}$
19:   for all $(\mathbf{h}^{j+\delta}_{j+\delta-n+2}, h_j, \gamma)$ in $\Gamma_{j+\delta}$ do
20:     $H_j \Leftarrow H_j \cup \{(h_j, \gamma)\}$
21:   end for
22:   $H_j \Leftarrow sumSame(H_j)$
23: end for
```

where γ is the value of $\gamma'(\mathbf{h}^{j-1}_{j-n+1}, h_i)$. In every step we compute for every possible $(\mathbf{h}^{j-1}_{j-n+1}, h_i, \gamma)$ in Γ_{j-1}, the new probability γ^*, and add $(\mathbf{h}^{j}_{j-n+2}, h_i, \gamma^*)$ to Γ'_j. The functions $sumSame(\Gamma'_i)$ and $trim(\Gamma'_i)$ (not shown in algorithm 2) perform the following operations on Γ' : $sumSame(\Gamma')$ sums the probabilities of all structures $(\mathbf{h}^{j}_{j-n+2}, h_i, \gamma)$ in Γ' that have the same value for $\mathbf{h}^{j}_{j-n+2}$ and for h_i. The function $trim(\Gamma', b)$ is the implementation of the $trim(\gamma'(\mathbf{h}^{j-1}_{j-n+1}, h_i))$ operator introduced earlier, i.e. it sorts all the structures according to decreasing γ, selects the b most likely values and rescales the counts by σ_{i-n+1} to make sure that the total probability mass remains 1.0.

Related work We are not the first authors discussing the high time complexity of HMM's, although none of the previous proposed methods could be used here. Mitchell et al. (1995) and Yu and Kobayashi (2003) propose algorithms that reduce the time complexity of second order explicit-duration HMM's, but the suggested algorithms are still quadratic in the number of hidden states, which is also the case in the on-line learning algorithm proposed by Krishnamurthy and Moore (1993). Shue and Dey (2002) develop a efficient algorithm for HMM's that have hidden states that are nearly completely decomposable, where the hidden states can be grouped together in "super-states", which is not the case here.

6.2.3 Training

During training we want to find parameters C and γ that optimize the likelihood of the model on an unseen test sequence. For this we use two corpora, a large corpus $\mathbf{w}_{train} = [w_1 ... w_{N_t}]$ and a smaller corpus $\mathbf{w}_{heldout} = [w_1 ... w_{N_h}]$. $\mathbf{w}_{train}$ is used to determine the counts C and $\mathbf{w}_{heldout}$ is used to determine the smoothing parameters γ.

We find the counts C that maximize the likelihood of the model on the training corpus $\mathbf{w}_{train}$ i.e. we select the parameters such that the model "explains" the observed words in the training corpus. In HMM's, this is typically performed using the expectation-maximization (EM) algorithm. The algorithm starts from a initial estimate of the counts C^1. This estimate is then improved in several iterations, where every iteration performs an expectation step and a maximization step. Since every step improves the likelihood of the model on $\mathbf{w}_{train}$, it is guaranteed to find a (possibly local) maximum likelihood estimate. In the expectation step the expected value of every hidden variable $P(h_i|\mathbf{w}_{train}, C^\tau, \gamma^\tau)$ is computed with the forward-backward algorithm.

For the LWLM, we perform three modifications to this algorithm. The first modification is that we first train a standard n-gram language model on the observed words of the training corpus. We then set the probability distribution of every hidden word h_i in the training corpus to $P(w_i|\mathbf{w}_1^{i-1}, \mathbf{w}_{i+1}^{N_t})$, i.e. to the probability of the observed word, given all words that occur before and after this word. This probability is computed as

$$P(w_i|\mathbf{w}_1^{i-1}, \mathbf{w}_{i+1}^{N_t}) \sim P(w_i|\mathbf{w}_{i-n+1}^{i-1}) \prod_{m=1}^{n-1} P(w_{i+m}|\mathbf{w}_{i+m+1}^{i+m+n-1})$$

This assignment makes the assumption that likely hidden words at a certain position are the words that are likely to be observed in that context, or more specifically, that are likely to be generated by the $n-1$ previous observed words, and that are likely to generate the $n-1$ next observed words. We assign these distributions to the hidden variable h_i on every position and then construct the initial counts C^1 from these distributions by collecting the following soft counts: (1) the frequency of the hidden word h_i generating the observed word w_i and (2) the frequency of the m-gram of hidden words $\mathbf{h}_{i-m+1}^{i}$ occurring in the sequence of hidden words,$\mathbf{h} = [h_1...h_N]$, where the length m ranges from n to 1.

The second modification to the Baum-Welch algorithm is the use of the forward-forward beam search to compute an approximation of $P(h_i|\mathbf{w}_{train}, C^{\tau}, \gamma^{\tau})$. The approximate values found by this method are used to construct the new counts $C^{\tau+1}$.[2]

The third modification is that we also update the smoothing parameters $\gamma^{\tau+1}$: we use an iterative line-search to find the parameters that optimize the likelihood of the parameters given the held-out corpus. After the smoothing parameters have been optimized, we again perform an iteration of the EM-algorithm. This cycle is repeated until the parameters have converged and the perplexity on the held-out corpus does not decrease anymore.

6.2.4 Predicting an unseen text

We have explained how we can estimate the hidden words for a particular given text. However, we would also like to use the model to predict the probability $P(\mathbf{w}_{test}|C, \gamma)$ of an unseen text $\mathbf{w}_{test} = [w_1...w_{N_u}]$. Standard HMM's use the forward algorithm to compute this value

$$P(\mathbf{w}_{test}|C, \gamma) = \sum_{\mathbf{h}_{N_u-n+2}^{N_u}} \alpha(\mathbf{h}_{N_u-n+2}^{N_u})$$

where $\alpha(\mathbf{h}_{i-n+2}^{i})$ is defined as in the previous section. This algorithm thus effectively passes a series of messages from the start of the sequence to the end. The messages $\alpha(\mathbf{h}_{N_u-n+2}^{N_u})$ at the end of the sequence are then summed, resulting in the probability of observing the entire sequence.

Since the time complexity of this algorithm is $O(N_u|V|^n)$, we propose the *forward beam search*. This method uses the trimmed $\alpha'(\mathbf{h}_{i-n+2}^{i})$ to compute

$$P(\mathbf{w}_{test}|C, \gamma) = \sum_{\mathbf{h}_{i-n+2}^{i}} \alpha'(\mathbf{h}_{i-n+2}^{i})$$

where $\alpha'(\mathbf{h}_{i-n+2}^{i})$ is defined as in the previous section. Note that, although this method drops unlikely $\alpha'(\mathbf{h}_{i-n+2}^{i})$ values, it assigns a non-zero probability to every possible sequence of observed words, thanks to the smoothing method used in equation 6.8. This method is outlined in algorithm 3.

[2]Note that in fact we also need to estimate $P(\mathbf{h}_{i-n+1}^{i}|\mathbf{w}_{train}, C^{\tau}, \gamma^{\tau})$, i.e. the probability of the observing the sequence $\mathbf{h}_{i-n+1}^{i}$ given the training text and the current parameters. For standard HMM's, this value can easily be computed using the forward and backward messages (see Bishop (2006)). An approximation to this value is also computed with the forward-forward beam search, we refer to section C in the appendix for details on this method.

Algorithm 3 Compute probability of observed text $\mathbf{w}_{test}$

```
Require: $\mathbf{w}_{test}$, $A_{N_t}$
Ensure: $P \simeq P(\mathbf{w}_{test}|C,\gamma)$
1: $P \Leftarrow 0$
2: for all $(\mathbf{h}^{j}_{j-n+2}, \alpha)$ in $A_{N_t}$ do
3:     $P \Leftarrow P + \alpha$
4: end for
```

N-gram models that map words to hidden variables or clusters often improve the ability of a model to predict the probability of unseen sequences, but they can also hurt precision when assigning to much weight to unseen sequences (Goodman, 2001). For this reason it is often a good idea to interpolate these models with a standard n-gram model. We define an interpolated version of the LWLM as

$$P(\mathbf{w}_{test}|C,\gamma) = \prod_{i=1}^{N_t} \left[\alpha P(h_i|\mathbf{h}^{i-1}_{i-n+1}, C, \gamma) \times P(w_i|h_i, C, \gamma) + (1-\alpha) P_{rkn}(w_i|w^{i-1}_{i-n+1})\right]$$

where $0 \leq \alpha \leq 1$ is a constant value optimized on a held-out text.

6.2.5 Implementation of the LWLM

We have discussed a number of adaptations to the standard algorithms used for HMM's to reduce the time complexity of these algorithms. However, even with this reduced complexity, a number of additional optimizations were implemented in order to be able to run the LWLM on a large dataset.

Distributed training We have developed a distributed computing infrastructure that enables the distributed computation of the LWLM on a large number of computers. This infrastructure is robust, very easy to set-up, downloads automatically the necessary class-files and datasets, and has remote exception handling. Furthermore it has been implemented for performance, with data caching, load balancing and automatically selection of the fastest computing clients. For the full description of this architecture we refer to section A in the appendix.

Additional optimization of the forward-forward and forward beam search We have seen how the time complexity of the forward-forward beam search is $O(N_t(b|V| + b|V|\log(b|V|))$ and the complexity of the forward beam search is $O(N_u(b|V| + b|V|log(b|V|))$. In fact, these algorithms are dominated by the $b|V|$ term, since sorting the values, with complexity of $b|V|log(b|V|)$, has a very small constant factor, and can be ignored in practice. Because of the large size of $|V|$, performing $b|V|$ computations makes the algorithm too slow for many applications. For this reason we only consider a subset $W \subset V$ of all hidden words. The possible hidden words h_i that are selected for a particular observed word w_i, are the words with highest values $P(w_i|h_i)P(h_i)$. We set the size of W greater then b, but substantially smaller than V, e.g. often we set $b = 50$ and $|W| = 200$. This optimization is used in the forward-forward beam search and the forward beam search. It is however not used when computing the likelihood of the model on a test text, since selecting W based on the observed word w_i when trying to predict this word, is a form of cheating, resulting in an unrealistic low likelihood.

6.3 Evaluating the proposed language models

In this section we evaluate the performance of the different language models. The performance of a language model is typically measured in terms of the perplexity of the model on the unseen test corpus $\mathbf{w} = [w_1...w_{N_u}]$

$$\sqrt[N_u]{\frac{1}{P(\mathbf{w}_T)}}$$

where $P(\mathbf{w}_T) = \prod_{i=1}^{N_u} P(w_i|\mathbf{w}_1^{i-1})$ is the probability of the test corpus. The perplexity can be seen as the confusion of the model. For example, a language model that assigns equal probability to 100 words at every position in the held-out corpus has a perplexity of 100 (assuming that at every position the observed word belongs to the set of predicted words). One attractive property of the perplexity measure is that the “true” model for any data source will have the lowest possible perplexity for that source. Thus, the lower the perplexity of our model, the closer it is in some sense, to the model of natural language employed by humans.

We first compare the performance of the latent words language model with a other language models in section 6.3.1 and then evaluate certain aspects of the proposed models in section 6.3.2.

6.3.1 Comparison

We compare the performance of the language models presented here on three different corpora. The first two corpora, *Reuters* and *APNews*[3], are collections of news articles that are distributed respectively by the Reuters and Associated Press news agencies. Both corpora have a large fraction of financial news, together

[3]We would like to thank Yoshua Bengio and Hugo Larochelle for providing this corpus.

with a smaller amount of general news. The third corpus *EnWiki* is a collection of encyclopedia articles from the English language Wikipedia. The *APNews* corpus was preprocessed as described by (Bengio et al., 2003), the other two corpora were preprocessed by concatenating all sentences and mapping all words that occurred less than 3 times in the combined training, held-out and test corpus to a new, "UNKNOWN" symbol. All punctuation was preserved. The vocabulary size of the corpora is 39373 for *Reuters*, 15247 for *APNews* and 54371 for *EnWiki*, reflecting the larger topical variation of the *EnWiki* corpus. For both corpora we use a fixed section of 100K consecutive words as a held out corpus, used to optimize the smoothing parameters of every methods, and a section of 100K consecutive words as a test corpus, used to measure the perplexity of every model.

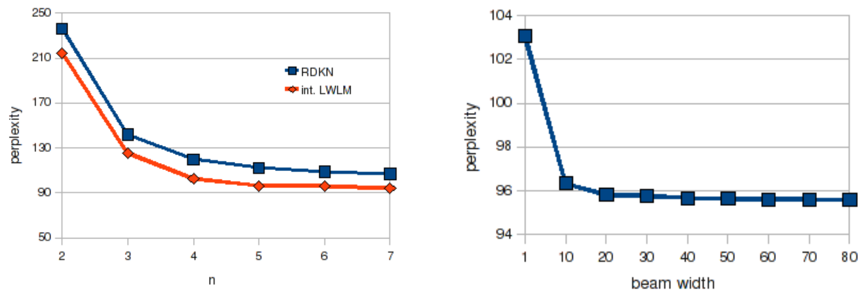
Table 6.1 shows the performance of the different models on the three corpora. We compare 5-gram models with interpolated smoothing (IP), absolute discounted Kneser-Ney smoothing (*ADKN*), relative discounted Kneser-Ney smoothing (*RDKN*), the latent words language model (*LWLM*) and the interpolated version of this model (*int. LWLM*). For comparison's sake we also include the results of the full-ibm-predict model (*IBM*), which is an existing language model that creates clusters of words that are syntactically and semantically similar, but contrary to our method, all words are hard-assigned to a single cluster (Goodman, 2001). It was found to be the best cluster-based model by Goodman (2001).

We make the following observations:

- Interpolated smoothing performs much worse then the other language models, which is well-known (e.g. Chen and Goodman (1996)).
- Relative discounted Kneser-Ney smoothing consistently outperforms absolute discounted Kneser-Ney smoothing. We have performed a fair comparison where both models used the same number of discount parameters. This refutes any optimality claims that have been made for absolute discounted Kneser-Ney smoothing (Chen and Goodman, 1996; Goodman, 2001). In fact we suspect that smoothing methods that use more complex methods for discounting will prove to be even more successful.
- The latent words language model outperforms both variants of Kneser-Ney smoothing. This shows that the algorithm successfully learned word similarities that alleviated the sparseness problems of n-gram models. We will discuss this in more depth in section 6.4.
- The interpolated latent words language model outperforms all other tested models. Compared to absolute discounted Kneser-Ney, which is a frequent used baseline, LWLM performs between 14% and 18% better. Compared to full-ibm-predict model, which is to our knowledge the best n-gram based language model, it performs between 7% and 10% better.

Method	ReutersNews	APNews	EnWiki
IP	130.61	148.49	170.29
IBM	108.38	125.65	149.21
ADKN	114.96	134.42	161.41
RDKN	112.37	132.99	160.83
LWLM	108.78	124.57	151.98
int. LWLM	**96.45**	**112.81**	**138.03**

Table 6.1: Results in terms of perplexity of the 5-gram models with interpolated (IP), absolute discounted Kneser-Ney (ADKN) or relative discounted Kneser-Ney (RDKN) smoothing, of the latent words language model (LWLM) and its interpolated version (int. LWLM) and of the *full ibm predict* class-based language model (Goodman, 2001).



(a) n-gram length n (b) beam size b

Figure 6.4: Perplexity of the interpolated LWLM, depending on the length of the n-gram (n) or of the beam size (b).

6.3.2 Additional experiments

In the following paragraphs we investigate certain properties of the interpolated LWLM. For all experiments we train the model on the $5M$ *Reuters* corpus.

In a first experiment we justify the choice for a relatively large value for n. Figure 6.4a shows how the perplexity of the model varies with increasing n. As n increases, we take into account a larger context, resulting a lower perplexity of the model. We see that we need to use a value of $n = 4$ or $n = 5$ for a competitive language model, justifying the approximate techniques developed in this chapter to make using such a large context window possible.

In a second experiment we see the influence of the *beam size* for the LWLM. For this we measure the perplexity of the interpolated LWLM with different beam

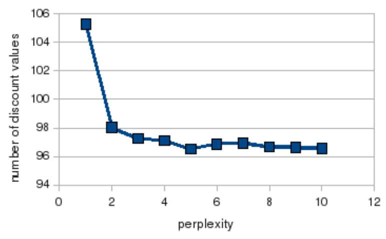


Figure 6.5: Perplexity of the interpolated LWLM for different number of parameters employed in the Kneser-Ney smoothing method.

sizes. We see from figure 6.4b that even with a beam size of 1, i.e. when we only consider the single most likely value for the hidden words, the LWLM outperforms RDKN (which achieves 112.37 on this corpus). With increased beam width the LWLM achieves lower perplexities, although this difference is small for beam sizes larger then 20. In our experiments we have chosen a conservative beam width of 50.

In a third experiment we see the influence of the *number of smoothing parameters* employed in our model. We use the definition given in section 6.1.4 for relative discounted Kneser-Ney smoothing: for the n-grams $h_{i-n+1}...h_i$ of a particular length n we create for the counts $c(h_{i-n+1}...h_i)$ a number of equally populated intervals. The counts in a particular interval are then discounted with a certain factor, unique to this interval. As we increase the number of intervals (and thus the number of discount factors), the model has more expressive power, most likely resulting in a more accurate model. Figure 6.5 shows how the perplexity of the model changes with the number of intervals employed in the discount function. We see that it is important to use more then one interval, but after that performance levels off quite quickly. In the experiments we set the number of intervals to 5. Note that for n-grams of different length, we have a different set of discount factors, e.g. for a model with $n = 5$ and 5 intervals for every discount function we have a total of 25 discount factors.

6.4 Semantic acquisition

A different method for assessing the performance of the LWLM is to perform a manual inspection of the hidden words that are estimated for a given sentence. An informal evaluation has shown that the discovered hidden words capture synonyms or related words, and that the hidden words are context dependent, essentially disambiguating the observed words. Table 6.2 gives the hidden words for an observed sentence from the training corpus, showing that for most words correct

a	**japanese**	**electronics**	**executive**	**was**	**kidnapped**	**in**	**mexico**
a	japanese	tobacco	executive	was	kidnapped	in	mexico
the	u.s.	electronics	director	is	abducted	on	usa
its	german	sales	manager	were	killed	at	uk
a	british	consulting	economist	are	found	of	australia
one	russian	electric	spokesman	be	abduction	into	canada

Table 6.2: Example of the most probable hidden words (bottom rows, sorted according to descending probability) for a given observed sentence (top row, bold) from the Reuters training corpus.

synonyms (e.g. kidnapped/abducted, executive/director) or related words (e.g. Japanese/u.s., abducted/killed) are found. This expansion can help to solve the underspecification and ambiguity problems of information extraction from natural language. In the next section we will see how indeed this model does improve semantic role labeling, especially in the context of a limited number of training examples.

6.5 Related work

Finding (soft) clusters of similar words, or finding similarities between words, has been a goal of NLP researchers for many years. It has been long time known that the similarity of two words could be learned by comparing their relative contexts in a large corpus: words occurring often in similar contexts tend to have a similar meaning. This was maybe most famously formulated as the distributional hypothesis, supported by theoretical linguists such as Harris (1954) and Firth (1957).

A large body of work has focused on methods to model the context of a particular word, and to compute a similarity measure based on these contexts. Frequent choices to model the context is a window of words or the direct dependents of the word in a syntactic dependency tree. Many methods have been used as similarity measures, ranging from the cosine measure, to the Jaccard index or the Kullback-Leibler distance (Pereira et al., 1993; Grefenstette, 1994; Lin, 1998b; Grishman and Sterling, 1994; Hearst, 1992).

Other researchers have focused on using generative models to learn classes of words. Related to our research is the work on unsupervised HMM's to learn a part-of-speech tagger (Merialdo, 1994). Typically a dictionary provides constraints in the form of possible part-of-speech tags for a large collection of words, which are used during the forward-backward algorithm to learn part-of-speech tags for all words in the corpus. Without these constraints, it is hard to learn accurate part-of-speech taggers (Smith and Eisner, 2005), although the importance of correct smoothing methods has also been recognized (Wang and Schuurmans, 2005). Recently, there has been some work on learning HMM's with Bayesian techniques such as Gibbs sampling (Goldwater and Griffiths, 2007; Johnson, 2007). These methods however could not be employed in this context, since they require impractically many (up to 20000) iterations to converge.

Other generative models were designed specifically for language modeling. Class-based language models aim to overcome the sparseness problems of n-gram language model by clustering all words in a large number of classes (Brown et al., 1992; Goodman, 2001). Typically a hard assignment is chosen, where a word belongs to exactly one class. Although these methods outperform standard Kneser-Ney smoothing, we have shown in the previous section that the LWLM outperforms these models in terms of perplexity on unseen texts. Furthermore, we will see in the next section that the probabilistic distances learned with the LWLM are more useful for improving information extraction methods than the classes.

Finally we would like to mention the interesting work performed by Collobert and Weston (2008) who propose a convolutional neural network architecture that is jointly trained for language modeling and for a number of different information extraction tasks. During the joint training, a look-up table is learned that maps words to a number of hidden classes. After training these classes represent syntactically and semantically related words, similar to the results achieved in our work. Another common point was that the authors found that a classifier for semantic role labeling achieved best results when trained jointly as a language model. In the next chapter we will see this is a result that is also confirmed by our work.

6.6 Conclusions of this chapter

In this chapter we have introduced the latent words language model. We started by answering the question "What happens if we replace the hidden states in a hidden Markov model with hidden words?". We saw that this model would have an infeasible large time complexity when computed exactly, and we have thus introduced a number of approximate methods with lower time complexity. The forward beam search was proposed as an approximate variant of the forward algorithm with lower time complexity. This algorithm was used in the forward-forward beam search to compute the expected value of the hidden words given an observed text, and in a method to compute the probability of an unseen text.

We have compared the LWLM with a number of other language models and seen that the interpolated LWLM outperforms all other n-gram models. We attribute this to the fact that the word similarities lessen the sparseness problem of traditional n-gram models. We also outperform the full-ibm-predict model, which learns hard word clusters. This can be explained by the fact that our probabilistic model is able to learn weighted similarities, i.e. it doesn't assume that a word is completely similar or dissimilar to another word.

Finally an informal inspection showed that the learned word similarities correlate with human assessment of similar words, and that the hidden words can be used to disambiguate a certain observed in a certain context. In the next chapter we will see how these can be used to augment supervised information extraction methods.

Chapter 7

Using unsupervised models for information extraction

In chapter 5 we argued that all weakly supervised models rely on the same assumption: examples that are close together in a high-density region have the tendency to be assigned the same label. In that chapter we have then proposed an approach where a single Bayesian network modeled both the space of examples and the labels assigned to these examples. Unlabeled data was then added to this model by considering the labels of the unlabeled examples as hidden variables that could be automatically estimated with Markov chain Monte Carlo methods.

In this chapter we discuss a different approach to weakly supervised learning: we use one unsupervised Bayesian network to model the space of examples and feed the structures learned by this model in a second Bayesian classifier that is used for classification. In this chapter we consider two models to model the input examples: the latent words language model and a class-based language model. We feed the results of these models in a supervised discriminative classifier. We test this approach on two information extraction tasks: word sense disambiguation and semantic role labeling.

7.1 Unsupervised models

We first train two unsupervised models on unlabeled data. The first model is the *latent words language model* (LWLM) discussed in the previous chapter. We train this model on a $20M$ words Reuters corpus to learn the counts C and smoothing parameters γ (see section 6.2.2). We then use this model to find the probability distribution $P(h_i|\mathbf{w}_1^{N_t}, C, \gamma)$ of hidden words h_i for every word w_i in the training and test data of the information extraction task at hand.

The second unsupervised model used in this chapter is the class-based language model *full ibm predict* (Goodman, 2001), which was also trained on the same $20M$ Reuters corpus. This model learns an assignment of every word in the vocabulary to a cluster c_i. Once this assignment is learned, we can trivially assign the correct cluster to every word in the training and test data for the information extraction task. The number of clusters is optimized on a held-out corpus and was selected to be 1250.

Both unsupervised models have exactly the same goal: minimizing the perplexity of the model on an unseen text. The main difference is that the first model learns a probabilistic mapping from the observed words to latent words, where the second model uses a hard mapping, where every observed word is assigned to a single cluster. A second important difference is the number of latent variables: for the LWLM the number of hidden variables is equal to the number of words in the vocabulary, while for the class-based language model the number of hidden variables is equal to the number of classes.

7.2 Words Sense Disambiguation

We first discuss weakly supervised learning for word sense disambiguation (WSD). WSD, was described in section 3.3 as the task of selecting the right sense of a particular word from a fine-grained dictionary of different senses depending on the context the word occurs in. We saw how the generative and discriminative models for this task use a large number of features extracted from the context to perform word disambiguation. In this section we will expand this model to the weakly supervised case where we incorporate structures learned from an unsupervised model.

7.2.1 Expanding the set of features

We first consider a method that employs the hidden words estimated by the LWLM. Given the hidden words for both the Semcor training set and the Senseval3 test set, we expand the standard set of features $\mathbf{f}(w_j)$ (section 4.3.1) with two types of probabilistic features. The first type is the hidden word for the word being disambiguated. $|V|$ extra values are thus appended to the feature vector $\mathbf{f}(w_j)$, containing the probability distribution over the $|V|$ possible values for the hidden variable h_i. The second type of features are the probability distributions for the hidden variables within a certain window of the current word. Also for these $|V|$ probability values are appended to the feature vector.

features	nouns	verbs	adjectives	all
standard	65.12	68.15	54.10	66.32
+hidden words	**67.36**	**69.35**	55.06	**67.61**
+clusters	66.5	68.59	**55.20**	66.97

Table 7.1: Results (in terms of % accuracy) for word sense disambiguation on the Senseval3 dataset using a supervised discriminative classifier with extra features derived from hidden words or clusters from a class-based language model.

We also test an alternative unsupervised model, the class-based IBM model. Also here two types of features are used: the class of the current word and the classes of all words within a certain window of the current word. These classes are all appended to the feature vector.

Once all feature vectors are expanded, they are used in a supervised discriminative classifier that is trained on the Semcor corpus.

7.2.2 Evaluation

The model is tested on the test data from the Senseval3 workshop (Snyder and Palmer, 2004). Also here the feature vectors are expanded as described above and used to test the accuracy of the model. Table 7.1 shows that both features improve the accuracy of the classifier. The accuracy for the classifier using hidden words features are 67.36%, 69.35% and 55.06% for nouns, verbs and adjectives respectively, which are all higher then for the supervised classifier. The accuracy of this classifier on all words is 67.61%, which is to the best of our knowledge, the best result achieved on this dataset.

The classifier using cluster features also outperforms the standard classifier, although it performs worse for nouns and verbs than the classifier that uses hidden words. Since nouns and verbs make up the majority of the words to be labeled, this classifier also performs lower overall.

7.3 Semantic role labeling

We consider two methods for weakly supervised semantic role labeling (SRL). In the first method the hidden word or hidden class are included as an extra feature (section 7.3.1). The second method automatically expands the training set by selecting similar sentences from a large unlabeled corpus (section 7.3.2). Both approaches are evaluated in section 7.3.3.

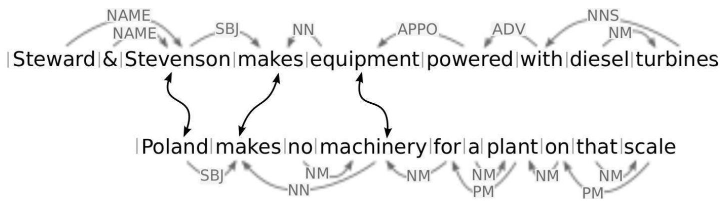

NAME
NAME
SBJ
NN
APPO
ADV
NNS
NM
Steward & Stevenson makes equipment powered with diesel turbines
Poland makes no machinery for a plant on that scale
SBJ
NM
NN
NM
NM
PM
NM
NM
PM

Figure 7.1: Mapping dependents of one occurrence of “makes” to another.

7.3.1 Expanding the set of features

First the hidden words $P(h_i|\mathbf{w}_1^{N_t}, C, \gamma)$ are estimated for both the CoNLL 2008 training and test set. Here we only use one type of probabilistic feature, the hidden words for the word that is currently being labeled. These probabilistic features are appended to the feature vector in a manner identical as for WSD.

We also test an alternative method where we append the feature vector with the cluster of the word being labeled, where the clusters are learned by the IBM class-based language model.

7.3.2 Automatic expansion of the dataset

In this section we discuss a different approach, where the training set is expanded with automatically labeled examples from a large unlabeled corpus. This method was first proposed by Fürstenau and Lapata (2009) and is tailored to the specific case of weakly supervised learning for SRL.

7.3.2.1 Original model

Given a set of labeled verbs with annotated semantic roles, Fürstenau and Lapata (2009) automatically find for every annotated verb similar occurrences of this verb in a large corpus of unlabeled texts. Given two occurrences of the same verb at position i with m dependents and at position j with n dependents, we define a mapping from i to j as an injective function $\sigma : M_i \rightarrow M_{\sigma(i)}$ that maps a non-empty subset $M_i \subset \{1, ..., m\}$ from the m dependents at position i to a non-empty subset $M_{\sigma(i)} \subset \{1, ..., n\}$ from the n dependents at position j , where $|M_i| = |M_{\sigma(i)}|$ and different dependents in the first occurrence are mapped to different dependents in the second occurrence.

For example, figure 7.1 shows two occurrences of the verb “makes”. For the first occurrence the verb has the direct dependents “Stevenson” and “equipment”, while in the second occurrence this verb has direct dependents “Poland” and "machinery". This results in a mapping from words "Stevenson" to "Poland" and from "equipment" to "machinery".

The similarity of this mapping is computed from the semantic and syntactic similarity between the mapped words and is given by

$$sim(\sigma) = \sum_{k \in |M_i|} \big(\alpha \cdot syn(w_k, w_{\sigma(k)}) + sem(w_k, w_{\sigma(k)}) - \beta\big) \tag{7.1}$$

Here α is a constant, weighting the importance of the syntactic similarity $syn(w_k, w_{\sigma(k)})$ compared to semantic similarity $sem(w_k, w_{\sigma(k)})$, and β can be interpreted as the lowest similarity value for which an alignment between two arguments is possible. $syn(w_k, w_{\sigma(k)})$ denotes the syntactic similarity between the dependency label of word w_k and the dependency label of word $w_{\sigma(k)}$. This value is defined as 1 if the dependency labels are identical, $0 < a < 1$ if the labels are of the same type but of a different subtype[1] and 0 otherwise. The semantic similarity $sem(w_k, w_{\sigma(k)})$ is estimated as the cosine similarity between the contexts of w_k and $w_{\sigma(k)}$ in a large text corpus.

The similarity between two occurrences of the same verb on positions i and j is defined by

$$sim(w_i, w_j) = \max_{\sigma} sim(\sigma) \tag{7.2}$$

We thus find the mapping σ with highest similarity that maps dependents of verb w_i to similar dependents of verb w_j and use this mapping to compute the similarity of the occurences at positions i and j. For every verb in the annotated training set we find the s occurrences of that verb in the unlabeled texts with the most similar contexts, given the best possible alignment. We then expand the training set with these examples, automatically generating an annotation using the discovered alignments. The variable s controls the trade-off between annotation confidence and expansion size. The final model is then learned by running the supervised training method on the expanded training set. The values for s, a, α and β are optimized automatically in every experiment on a held-out set (disjoint from both training and test set).

7.3.2.2 Including hidden words

We adapt this approach by employing a different method for measuring semantic similarity. Given two words w_i and $w_{\sigma(i)}$ we estimate the probability distribution of

[1]Subtypes are fine-grained distinctions made by the parser such as the underlying grammatical roles in passive constructions.

the hidden word on these positions, which we refer to with $H(h_i) = P(h_i|\mathbf{w}_1^{N_t}, C, \gamma)$ and $H'(h_{\sigma(i)}) = P(h_{\sigma(i)}|\mathbf{w}_1^{N_t}, C, \gamma)$. We use the Jensen-Shannon (Lin, 1997) divergence to measure the distance between these two distributions, given by

$$JS(H(h_i)||H'(h_{\sigma(i)})) = \frac{1}{2}\left[D\left(H(h_i)||avg\right) + D\left(H'(h_{\sigma(i)})||avg\right)\right] \tag{7.3}$$

where $avg = \frac{H(h_i)+H'(h_{\sigma(i)})}{2}$ is the average between the two distributions and $D\left(H(h_i)||avg\right)$ is the Kullback-Leibler divergence (Cover and Thomas, 2006) given by

$$D\left(H(h_i)||avg\right) = \sum_{h_i} H(h_i) log(\frac{H(h_i)}{avg(h_i)}) \tag{7.4}$$

The Jensen-Shannon divergence is a positive number greater than or equal to 0 that is closer to 0 if the two distributions are more similar. This divergence is converted to a similarity value between 0 and 1 with

$$sim_1(w_i, w_{\sigma(i)}) = exp(-\lambda \times JS(H(h_i)||H'(h_{\sigma(i)}))) \tag{7.5}$$

here λ is a constant that is optimized on a held-out set.

We also experiment with a similarity measure proposed by Lin (1998a). This author defines similarity between two distributions as the ratio of the information shared by the two distributions and the information in every distribution separately. In our case this translates to

$$sim_2(w_i, w_{\sigma(i)}) = \frac{2 \times \sum_{h_i} H(h_i)H'(h_i)log(P(h_i))}{\sum_{h_i} H(h_i)log(P(h_i)) + \sum_{h_i} H'(h_i)log(P(h_i))} \tag{7.6}$$

where $P(h_i)$ is unigram probability of the hidden word h_i, which is independent of H or H'. In this similarity measure infrequent hidden words carry a relative higher weight then frequent hidden words. This weight is proportional to the information content $log(P(h_i))$ of $P(h_i)$.

We adapt the original expansion algorithm with these two similarity measures. Although these changes might appear only a slight deviation from the original model discussed by Fürstenau and Lapata (2009) it is potentially an important one, since an accurate semantic similarity measure will greatly influence the accuracy of the alignments, and thus of the accuracy of the automatic expansion.

7.3.3 Evaluation of weakly supervised SRL

We perform a number of experiments where we compare the standard supervised discriminative model with the different weakly supervised methods proposed in sections 7.3.1 and 7.3.2.

	5%	20%	50%	100%
Discriminative	40.49%	67.23%	74.93%	78.65%
HWFeatures	**60.29%**	**72.88%**	**76.42%**	**80.98%**
ClassFeatures	59.51%	66.70%	70.15%	72.62%
CosExp	47.05%	53.72%	64.51%	70.52%
JSExp	45.40%	53.82%	65.39%	72.66%
LinExp	51.84%	57.98%	67.39%	74.66%

Table 7.2: Results (in F1-measure) on the CoNLL 2008 test set, comparing the standard supervised classifier with different weakly supervised classifiers, using different portions of the full training set for training. See main text for details.

HWFeatures Add the hidden words as probabilistic features.

ClassFeatures Add the class from a class-based language model as extra feature.

CosExp Expand training set, use cosine for semantic distance.

JSExp Expand training set, use Jensen-Shannon divergence on hidden words.

LinExp Expand training set, use Lin's distance measure on hidden words.

Table 7.2 shows the results of the different supervised and semi-supervised methods on the test set of the CoNLL 2008 shared task. We experimented with different sizes for the training set, ranging from 5% to 100%. When using a subset of the full training set, we run 10 different experiments with random subsets and average the results.

We see that the *HWFeatures* method performs better than the other methods across all training sizes. Furthermore, these improvements are larger for smaller training sets, showing that the approach can be applied successfully in a setting where only a small number of training examples is available. When comparing the *HWFeatures* method with the *ClusterFeatures* method we see that, although the *ClusterFeatures* method has a similar performance for small training sizes, this performance drops for larger training sizes. A possible explanation of this result is the use of the clusters employed in the *ClusterFeatures* method. By definition the clusters merge many words into one cluster, which might lead to good generalization (more important for small training sizes) but can potentially hurt precision (more important for larger training sizes).

We compare the different methods for automatic expansion (*CosExp*, *JSExp* and *LinExp*) to the supervised classifier and see that all three methods have improved performance for small training sizes, but reduced performance for larger training sizes. An informal inspection showed that for some examples in the training set, little or no correct similar occurrences were found in the unlabeled text. The algorithm however always adds the most similar s occurrences to the training set for every annotated example, also for these examples where little or no similar occurrences were found. In these cases the automatic alignment often fails to generate correct labels and introduces errors in the training set. In the future we plan experiments that determine dynamically (e.g. based on the similarity measure between occurrences) for every annotated example how many training examples to add.

7.3.4 Related work

The most popular unsupervised models used for information extraction are clustering methods that learn clusters of semantic and syntactic similar words. These clusters are then used as extra features in the information extraction task. Tang et al. (2001) uses the clusters from the class-based Brown language model (Brown et al., 1992) in sentence parsing, but do not compare their classifier to a classifier without these clusters. These clusters are also used by Koo et al. (2008) in a syntactic dependency parser with an error reduction of 14.29% compared to a parser without these clusters, and by Zhao et al. (2009) in a multilingual dependency parser. Miller et al. (2004) use hierarchical word clusters (optimized for bigram perplexity) in a discriminative named entity recognizer, achieving a 25% error reduction compared to a classifier without these clusters.

We have shown that for WSD and SRL using the LWLM hidden words improves accuracy compared to the clusters from a class-based language model, and we can thus safely assume that this will also hold for these other information extraction tasks.

The research closest to our work is Li and McCallum (2005), who use the soft clusters derived by Griffiths et al. (2005) in a supervised conditional random field classifier for part-of-speech tagging and Chinese word segmentation, with an 14% error reduction compared to a classifier without these features.

7.4 Conclusions

In this chapter we have discussed a different approach to weakly supervised learning: feed the structures learned in an unsupervised model into a supervised model trained for a certain information extraction task. We have discussed two methods: use the hidden words or clusters as extra features and use these hidden words and clusters in a similarity measure to automatically expand the training set.

For the second approach we have performed a number of experiments where we use a similarity measure based on a cosine distance, or on a distance metric that uses the hidden words with the Jensen-Shannon divergence or Lin's similarity measure. We saw that these methods only outperformed the supervised classifier for small training sets, because of the incorrect examples introduced by the automatic expansion.

Using the hidden words or clusters was a better approach that resulted in a significant improvement over the supervised classifier for both word sense disambiguation and semantic role labeling. This improvement was largest for small training sets, showing that this method does successfully reduce the dependency of the supervised model on large training corpora. Additionally this method also outperformed a method that uses clusters from a class-based language model. Because of its simplicity and independence of the specific information extraction task, we expect that this method can be employed almost effortless in other tasks, such as named entity recognition or part-of-speech labeling.

Part III

Automatic annotation of images and video

Outline part III : Information extraction for weak supervision of images and video

In the previous chapters we have discussed various weakly supervised methods and we have shown that these methods can be successfully employed to improve information extraction methods. We have however only considered labeled and unlabeled *textual data*, i.e. data in a single medium. We now turn to the case of *multimodal weakly supervised learning*. In this part we discuss methods that employ information extraction methods to aid the automatic analysis of images and video.

In **chapter** 8 we develop the *appearance model* which finds the entities present in an image by analyzing a text describing this image. This model is subsequently used in two applications, to align names in the text with faces in the image, and to perform textual image retrieval.

Chapter 9 deals with the automatic annotation of video. We first focus on the automatic annotation of actions of actors in the video, and apply the previously developed semantic role labeling system to the transcripts of a video series. In a second task we combine information extracted from the transcript with an automatic analysis of the video to discover the different scenes in a video, and to derive the location for every scene.

This research is motivated by the observation that frequently, the difficulties faced by automatic methods for image analysis are even greater than these faced by natural language processing methods, because of the large variations in scale, lightning conditions and relative orientation of entities in images.

The work in this part of the thesis is described in the following articles:

- Koen Deschacht and Marie-Francine Moens. *Text Analysis for Automatic Image Annotation.* In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, Prague, 2007.
- Koen Deschacht and Marie-Francine Moens. *Finding the Best Picture: Cross-Media Retrieval of Content.* In C. Macdonald, I. Ounis, V. Plachouras & I. Ruthven (Eds.) Proceedings of the 30th European Conference on Information Retrieval. Lecture Notes in Computer Science 4956 (pp. 539-546), Springer, 2008.
- Koen Deschacht, Marie-Francine Moens and Wouter Robeyns. *Cross-Media Entity Recognition in Nearly Parallel Visual and Textual Documents.* In Proceedings of the 8th RIAO conference on Large-Scale Semantic Access to Content (Text, Image, Video and Sound), USA, 2007.

- Erik Boiy, Koen Deschacht and Marie-Francine Moens. *Learning Visual Entities and their Visual Attributes from Text Corpora*. In Proceedings of the 5th International Workshop on Text-based Information Retrieval, IEEE Computer Society Press, 2008.
- Koen Deschacht and Marie-Francine Moens. Text Analysis for Automatic Image Annotation, In Proceedings of the 19th Belgian-Dutch Conference on Artificial Intelligence (Dastani, M. and de Jong, E., eds.), pp. 260-267, The Netherlands, 2007.

Chapter 8

Automatic annotation of images

In this chapter we describe methods for the analysis of texts that describe an image, with the goal of automatically creating annotations of images. This work is motivated by fact that information extraction methods on images face even greater difficulties than methods on text: entities for example often have dramatically different appearances in different images, depending on pose, lighting conditions, distance to the subject and other factors.

We start by describing the appearance model for entities in images (section 8.1) and then extend this model to attributes (section 8.2). In section 8.3 we compare this model to related research and we see in section 8.4 how it can be used in two applications: name and face alignment (section 8.4.1) and image retrieval (section 8.4.2). We conclude in section 8.5.

8.1 An appearance model for entities

Figure 8.1 shows an example of an image-text pair, where the text describes the entities that are present in the image. In this chapter we want to develop an automatic method that can determine from this descriptive text that e.g. "Hillary Clinton", "Bill Clinton", "David Paterson" and "Eliot Spitzer" are entities that are likely to appear in the image. We limit this analysis to text only, without considering any information present in the images.

To solve this task we propose the *appearance model.* This model assigns to every entity in the text a probability of this entity being present in the image.

New York State Sen. Hillary Clinton celebrates after making her nomination acceptance speech along with husband and former U.S. President Bill Clinton, left, David Paterson, second right, and gubernatorial candidate Eliot Spitzer, right, during the New York State Democratic Convention in Buffalo, N.Y., on Wednesday, May 31, 2006. Paterson is running for the New York State Lt. Governor's office.

Figure 8.1: Example image-text pair.

We first detect all entities (section 8.1.1), and determine for every entity its visualness (section 8.1.2) and salience (section 8.1.3), which are then combined in the appearance model (section 8.1.4). Finally we evaluate our model in section 8.1.5.

8.1.1 Entity detection

To detect all entities in the text we rely on existing tools for named entity recognition and part-of-speech tagging.

We use an existing named entity recognition package to recognize person names in the text. The OpenNLP package[1] detects noun phrase chunks in the sentences that represent persons, locations, organizations and dates. To improve the recognition of person names, we use a dictionary of names, which we have extracted from the Wikipedia[2] website. The noun phrase co-referents in the texts that are in the form of pronouns (e.g. "he", "she") are resolved with the LingPipe[3] package.

We use LTPOS (Mikheev, 1997) to perform part-of-speech tagging (i.e., detecting the syntactic word class such as noun, verb, etc.) and assume that every noun phrase in the text represents an entity.

8.1.2 Visualness

Any given text contains a large number of entities. We first develop a soft filter that assign to every entity in the text a probability that this entity is *visual*, e.g. that it can be perceived visually. Example of entities that can be perceived visually

[1]http://opennlp.sourceforge.net/
[2]http://en.wikipedia.org/
[3]http://www.alias-i.com/lingpipe/

are “car” and “house”, while “agreement” or “thought” can not be expected to be directly perceived visually.

We first determine the meaning of all entities in the text with respect to the WordNet database and then employ a distance metric together with a number of seed synsets to compute the visualness.

Word sense disambiguation After we have performed entity detection, we want to classify every entity according to the WordNet semantic database (Fellbaum, 1998). We use the word sense disambiguation (WSD) system described in chapter 3. Additionally we assign all person names detected by the named entity recognizer to the synset that corresponds to a human being.

WordNet similarity We determine the visualness for every synset using a method that is inspired by Kamps and Marx (2002). Kamps and Marx use a distance measure defined on the adjectives of the WordNet database together with two seed adjectives to determine the emotive or affective meaning of any given adjective. They compute the relative distance of the adjective to the seed synsets “good” and “bad” and use this distance to define a measure of affective meaning.

We take a similar approach to determine the visualness of a given synset. We first define a similarity measure between synsets in the WordNet database. Then we select a set of seed synsets, i.e. synsets with a predefined visualness, and use the similarity of a given synset to the seed synsets to determine the visualness.

Distance measure The WordNet database defines different relations between its synsets. An important relation for nouns is the hypernym/hyponym relation. A noun X is a hypernym of a noun Y if Y is a subtype or instance of X. For example, “bird” is a hypernym of “penguin” (and “penguin” is a hyponym of “bird”). A synset in WordNet can have one or more hypernyms. This relation organizes the synsets in a hierarchical tree (Hayes, 1999).

The similarity measure defined by Lin (1998b) uses the hypernym/hyponym relation to compute a semantic similarity between two WordNet synsets S_1 and S_2. It is based on the intuition that entities close in the hypernym tree should have a high similarity. The method first finds the most specific (lowest in the tree) synset S_p that is a parent of both S_1 and S_2. Then it computes the similarity of S_1 and S_2 as

$$sim(S_1, S_2) = \frac{2logP(S_p)}{logP(S_1) + logP(S_2)}$$

Here the probability $P(S_i)$ is the probability of labeling any word in a text with synset S_i or with one of the descendants of S_i in the WordNet hierarchy. We estimate these probabilities by counting the number of occurrences of a synset in the Semcor corpus (Fellbaum, 1998; Landes et al., 1998), where all noun chunks are labeled with their WordNet synset. The probability $P(S_i)$ is computed as

$$P(S_i) = \frac{C(S_i)}{\sum_{n=1}^{N} C(S_n)} + \sum_{k=1}^{K} P(S_k)$$

where $C(S_i)$ is the number of occurrences of S_i, N is the total number of synsets in WordNet and K is the number of children of S_i. The `WordNet::Similarity` package (Pedersen et al., 2004) implements this distance measure and was used by the authors.

Seed synsets We have manually selected 25 seed synsets in WordNet, where we tried to cover the wide range of topics we were likely to encounter in the test corpus. We have set the visualness of these seed synsets to either 1 (visual) or 0 (not visual). We determine the visualness of all other synsets using these seed synsets. A synset that is close to a visual seed synset gets a high visualness and vice versa. We choose a linear weighting:

$$vis(s) = \sum_i vis(s_i) \frac{sim(s, s_i)}{C(s)}$$

where $vis(s)$ returns a number between 0 and 1 denoting the visualness of a synset s, s_i are the seed synsets, $sim(s,t)$ returns a number between 0 and 1 denoting the similarity between synsets s and t and $C(s)$ is constant given a synset s:

$$C(s) = \sum_i sim(s, s_i)$$

8.1.3 Salience

Not all entities discussed in a text are equally important. We would like to discover what entities are in the focus of a text and what entities are only mentioned briefly, because we presume that more important entities in the text have a larger probability of appearing in the image than less important entities. We define the *salience* measure, which is a number between 0 and 1 that represents the importance of an entity in a text. We present here a method for computing this score based on an in depth analysis of the discourse of the text and of the syntactic structure of the individual sentences.

8.1.3.1 Discourse segmentation

The discourse segmentation module, which we developed in earlier research, hierarchically and sequentially segments the discourse in different topics and subtopics resulting in a table of contents of a text (Moens, 2008). The table shows the main entities and the related subtopic entities in a tree-like structure that also indicates the segments to which an entity applies. The algorithm detects patterns of thematic progression in texts and can thus recognize the main topic of a text (i.e., about whom or what the text speaks) and the hierarchical and sequential relationships between individual topics. A mixture model, taking into account different discourse features, is trained with the Expectation Maximization algorithm on an annotated DUC-2003 corpus. We use the resulting discourse segmentation to define the salience of individual entities that are recognized as topics of a sentence. We compute for each noun entity e_r in the discourse its salience ($Sal1$) in the discourse tree, which is proportional with the depth of the entity in the discourse tree -hereby assuming that deeper in this tree more detailed topics of a text are described- and normalize this value to be between zero and one. When an entity occurs in different subtrees, its maximum score is chosen.

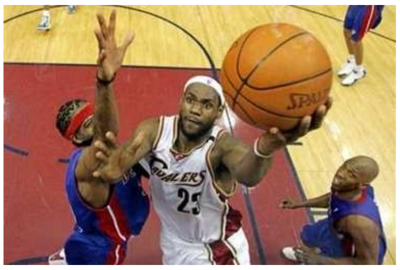
8.1.3.2 Refinement with sentence parse information

The segmentation module already determines the main topic of a text. Since the syntactic structure is often indicative of the information distribution in a sentence, we can determine the relative importance of the other entities in a sentence by relying on the relationships between entities as signaled by the parse tree. When determining the salience of an entity, we take into account the level of the entity mention in the parse tree ($Sal2$), and the number of children for the entity in this structure ($Sal3$), where the normalized score is respectively inversely proportional with the depth of the parse tree where the entity occurs, and proportional with the number of children.

We combine the three salience values ($Sal1$, $Sal2$ and $Sal3$) by using a linear weighting.

$$sal(e_i) = \alpha_1 Sal1 + \alpha_2 Sal2 + \alpha_3 Sal3$$

We determine coefficients for these three values on a held-out corpus, and set them to $\alpha_1 = 0.8$, $\alpha_2 = 0.1$ and $\alpha_3 = 0.1$.



Cleveland Cavaliers' LeBron James (23) shoots between Detroit Pistons' Richard Hamilton, left, and Chauncey Billups late in the fourth quarter of the Pistons' 84-82 win in a second-round NBA playoff basketball game Friday, May 19, 2006, in Cleveland. The series is tied at three games each.

LeBron James	0.83	Chauncey Billups	0.39	Richard Hamilton	0.31
Cavaliers	0.17	Cleveland	0.13	Pistons	0.10

Figure 8.2: Image-text pair with automatic extracted entities and predicted probabilities.

8.1.4 Appearance model

The appearance model for entities combines the visualness and salience measures. We want to calculate the probability of the occurrence of an entity e_{im} in the image, given a text t, $P(e_{im}|t)$. We assume that this probability is proportional with the degree of visualness and salience of e_{im} in t. In our framework, $P(e_{im}|t)$ is computed as the product of the salience of the entity e_{im} and its visualness score, as we assume both scores to be conditionally independent, given the value of e_i.

$$app(e_i) = sal(e_i) \times vis(e_i)$$

8.1.5 Evaluation of the appearance model

In this section we evaluate our approach on a real world corpus of near-parallel image-texts pairs.

8.1.5.1 Data set

We use a parallel corpus[4] consisting of 100 images-text pairs that were randomly selected out of a larger corpus of 1700 text pairs. The images and their captions

[4]We thank Yves Gufflet from the INRIA research team (Grenoble, France) for collecting this dataset.

are retrieved from the Yahoo! News website[5] and are similar to the “Faces in the wild” benchmark corpus (Huang et al., 2007). The captions will in general discuss one or more persons in the image, possibly one or more other objects, the location and the event for which the picture was taken. An example of an image-text pair is given in fig. 8.2. Not all persons or objects who are pictured in the images are necessarily described in the texts and vice versa.

Every image-text pair is annotated by one annotator, who has labeled the entities (i.e. persons and other objects) that appear in the text, the entities that appear in the image and the entities that appear in both. Fig. 8.2 shows an example image-text pair, where "Lebron James" “Chauncey Billups” and “Richard Hamilton” are the only entities that appear both in the text and in the image. On average the texts contain 15.04 entities, of which 2.58 appear in the image.

8.1.5.2 Experiments

We test a number of methods on the Yahoo! News corpus (table 8.1), using combinations of the methods discussed. A first method (*Ent*) uses only the entity detection, achieving a precision of 15.62% and recall of 91.08%. The low precision is caused by the fact that many entities in the texts are not visible in the image. Although these results indicate that this is a very naive baseline, it is in fact used quite often (see section 8.3). The second test (*Ent+Vis*) uses the selected entities together with a static cut-off value[6] on the visualness measure, achieving a precision of 48.81% and a recall of 87.98%. Although this method is already more successful in selecting the right entities is still suffers from the problem that also entities in the text that could be present on the image, are not necessarily so.

The third method (*Ent+Sal*) uses the entity selection together with a cut-off on the salience measure, which results in 66.03% precision and 54.26% recall. This shows that also the salience measure alone is not sufficient to select the correct entities. Our final method (*Ent+Vis+Sal*) combines entity detection with a static cut-off value of the combined visualness and salience measures. This method achieves 70.56% precision and 67.82% recall, which is the best result of the evaluated systems, both in terms of precision and f1-measure (69.39%).

Although the presented methods are quite successful in annotating the images *without an analysis of the images themselves*, it is interesting to see what are the most important factors for the incorrect annotations. We have manually evaluated the performance of the different techniques on this corpus. Both named entity recognition and part-of-speech tagging were quite accurate with 93.37% and 98.14% precision and 97.69% and 97.36% recall respectively. The visualness

[5]http://news.yahoo.com/

[6]All cut-off values in this section were manually selected based on a small set of held-out annotated examples.

	precision	**recall**	**F1-measure**
Ent	15.62%	91.08%	26.66%
Ent+Vis	48.81%	87.98%	62.78%
Ent+Sal	66.03%	54.26%	59.56%
Ent+Vis+Sal	70.56%	67.82%	69.39%

Table 8.1: Evaluation of different methods for automatically annotating entities in images, using entity recognition (Ent), the visualness (Vis) and salience (Sal) measures, and combinations hereof.

"African violets (Saintpaulia ionantha) are small, flowering houseplants or greenhouse plants belonging to the Gesneriaceae family. They are perhaps the most popular and most widely grown houseplant. Their thick, fuzzy leaves and abundant blooms in soft tones of violet, purple, pink, and white make them very attractive. Numerous varieties and hybrids are available. African violets grow best in indirect sunlight."

Figure 8.3: Example section of the *plants* corpus.

measure (with static cut-off) has an accuracy of 79.56%, where the errors are mainly caused by incorrect word sense disambiguation (63.10%) and in a lesser extent by the distance measure (36.90%). We did not evaluate the salience measure, since it is not trivial to exactly pin-point the most important entities in a text. For an evaluation of the discourse segmentation module we refer to (Moens, 2008).

8.2 A corpus based visualness measure

The work described in this section is joint work with Erik Boiy and Marie-Francine Moens.

We extend the visualness measure defined in the previous section. We make two extensions, first we perform a study of techniques to compute this visualness using corpus based association techniques. Secondly we compute the visualness of entities and attributes, where attributes are usually expressed by adjectives, such as "white", "small", and "wooden". We first discuss corpus based association techniques (section 8.2.1) and an extension of the previously defined WordNet similarity for adjectives (section 8.2.2). These two techniques are then combined (section 8.2.3) and evaluated (section 8.2.4).

> "Rebirth refers to a process whereby beings go through a succession of lifetimes as one of many possible forms of sentient life, each running from conception to death. It is important to note, however, that Buddhism rejects concepts of a permanent self or an unchanging, eternal soul, as it is called in Christianity or Hinduism. [...]"

Figure 8.4: Example section of the *religion* corpus.

8.2.1 Corpus-based association techniques

Assocation techniques provide methods to decide whether two observations occur more frequently together then would be expected due to chance. Popular measures are the chi-square metric and the likelihood ratio (Dunning, 1993). Several researchers have used these techniques to find word collocations (e.g. "the red cross", "the white house") (Dunning, 1993; Smadja, 1994), for automatic lexicon construction (Roark and Charniak, 1998) or for classification of words along a certain dimension (Turney, 2002).

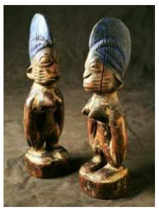
We have downloaded a collection of descriptions of the appearances of flowers and plants, which can be considered to contain mostly visual entities and attributes (*plants* corpus, figure 8.3) and all articles in the English wikipedia on religion, which can be considered to contain mostly non-visual entities (*religion* corpus, figure 8.4). We then use the χ^2-test (Chernoff and Lehmann, 1954) to decide whether a word is should be considered visual or not. More specifically we compute the χ^2 value and consider all words (both nouns and adjectives) above a certain threshold to be visual.

8.2.2 WordNet similarity for adjectives

Previously we have used WordNet to compute the visualness measure for entities. For adjectives we use a similar approach but have to use a different distance measure since WordNet does not define a hypernym/hyponym relation between adjectives. In stead we use the similarity measure of Lesk (1986), where the similarity between two synsets is defined as the overlap between the words in the textual descriptions of these synonyms. As for the entities we pick 25 seed adjectives and manually set their visualness to 0 or 1.

8.2.3 Combining association techniques and WordNet distance

The corpus based association metric gives a list of adjectives and nouns that are ranked according to χ^2-value, where words that have are higher ranked can be



"These small sculptures depict two identical human figures. The wooden bodies are weathered brown and the hair is faded blue. Both sculptures have a round base about one inch high. The feet are large and flat, with grooves cut into the front to distinguish toes. The legs are short [...]"

Figure 8.5: Example of the *art* corpus.

expected to be more visual. We use this fact to automatically select the seed synsets used for the WordNet similarity measure. For both nouns and adjectives we select the 13 highest (e.g. with large, positive χ^2 value, thus having a positive correlation) and 12 lowest ranked synsets (e.g. with low, negative χ^2 value, thus having a negative correlation) and set their visualness respectively to 1 and 0. The seeds are thus chosen automatically, making the visualness measure an unsupervised metric.

8.2.4 Evaluation of corpus based visualness

We evaluate the proposed techniques on a third corpus, that contains a mix of visual and non-visual entities and attributes. The corpus consists of a collection of descriptions of works of art together with an elaborate history of the object and the artist, and will henceforth be known as the art corpus (figure 8.5). A collection of these descriptions are manually annotated where every attribute and entity is labeled as visual or non-visual.

For every technique we manually set a static cut-off value on a small number of held-out descriptions. Table 8.2 shows the results for the different methods. We first see that the corpus based method performs much better for attributes than for entities, which is mainly caused by a low recall on the entities. We hypothesize that this can be attributed to the fact that attributes are more generic and can thus more easily be transferred to a different corpus. A second observation that can be made from table 8.2 is that the WordNet based method outperforms the corpus based method for entities but performs worse for attributes. Finally we see that the automatic selection of seed sets *improves* the results of the WordNet based method.

	precision		**recall**		**F1-measure**	
	attribs	ents	attribs	ents	attribs	ents
corpus	88.26%	81.71%	80.15%	50.95%	84.01%	62.76%
wordnet	81.13%	82.02%	50.15%	62.02%	61.98%	70.63%
combination	87.80%	82.67%	53.25%	66.22%	66.30%	73.54%

Table 8.2: Evaluation of different methods for automatically annotating entities (*ents*) and attributes (*attribs*) in images, using association techniques on (*corpus*), WordNet similarity (*wordnet*) and the combination of these techniques (*combination*).

8.3 Related Research

In recent years many researchers have worked on combining information found in images and associated texts. We limit our review to research that uses associated natural language texts such as captions or transcripts, ignoring approaches that use manually annotated keywords, since we are interested in methods to automatically select words that describe the image.

Named entity recognition is used by researchers interested in combining names in texts with faces in images. Most of these researchers assume that all persons are equally likely to appear in the image (Yang et al., 2004; Ozkan and Duygulu, 2006; Guillaumin et al., 2008). Other researchers acknowledge that this can be improved by having a measure that captures how likely people to appear in the image. Yang et al. (2005) select only persons that perform a monologue speech since these are more likely to appear in the video, while Satoh et al. (1999) use a rudimentary approach to discourse analysis that takes into account the position of the person in the transcript and the verbs that co-occur with this person, where a small manual selection of verbs is given a high score and all other verbs a lower score. Regretfully these two publications do not evaluate their approaches. Berg et al. (2004) construct a more elaborate context model to determine which persons appear in the image. This model includes the part-of-speech tags on both sides of the person name, the distances to the nearest special token (“,”, “.”, “(“, “)”, “(L)”, “(R)” or “(C)”), and the location of the name in the caption. The parameters of this model are then learned in an unsupervised manner, which results in a classifier that has an accuracy of 84%, where a baseline approach that assumes that all persons appear in the image achieves an accuracy of 67%. Contrary to our research this method takes into account highly corpus specific context cues and does not consider objects other then persons. Furthermore errors in the named entity recognition system are not taken into account in the evaluation, although the previously reported accuracy of the employed recognizer was between 80% and 90% (Cunningham et al., 2002).

Other researchers do not limit the text analysis to person names, although also here typically word selection is considered a preprocessing step and is given little attention. Jain et al. (2007) for example do not perform word selection, Mori et al. (2000) select nouns and adjectives when they occur above a certain frequency in the entire corpus, Westerveld (2000) lemmatises all words in the captions and uses words that occur at least in two different documents, Amir et al. (2005) perform stop-word removal and Porter stemming and assign $tf * idf$ weights to the remaining words and Westerveld et al. (2005) use a (retrieval) language modelling approach that interpolates a ML model for the text associated with the shot, with one associated to the scene, the video and the collection. None of these publications evaluate word selection separately.

Following our publications (Deschacht and Moens, 2007; Deschacht et al., 2007) we have seen more research on this task. Kliegr et al. (2008) perform entity detection in a similar manner as presented here, however a more advanced method is employed to map person names to WordNet. Where we mapped all person names to the synset representing a "human being", the authors perform a more fine-grained mapping where for example "David Beckham" is mapped to "footballer". This is accomplished by automatically learning hypernym relationships from a large corpus using Hearst-style patterns (Hearst, 1992), with an accuracy of 85%. The synsets are then manually classified according to visualness. Leong and Mihalcea (2009) have adapted our work to automatically annotate all elements of an image, and not only entities. They learn a visualness measure from a large corpus and modify the salience measure to include two other clues: semantic cloud, which captures which words are more "central" to a certain topic and lexical distance, which is the distance of words to the image. They achieved an $F1$-measure of 54.21% on a corpus of 180 images and corresponding web pages, compared to a $tf * idf$ baseline of 41.48%. Xia et al. (2009) aim at annotating images with full sentences. They use the visualness measure together with a latent semantic analysis of words and features extracted from the image to generate a set of candidate keywords for a given image. From a large corpus, all sentences that contain at least two of these keywords are ranked according to frequency on the world wide web. The accuracy of their method was not evaluated.

8.4 Applications of the appearance model

In this section we see two applications of the appearance model. We will see how it can be used to align names and faces (section 8.4.1) and to improve image retrieval (section 8.4.2).

8.4.1 Alignment of names and faces

The work described in this section was performed by Phi The Pham, Marie-Francine Moens and Tinne Tuytelaars.

One of the goals of the appearance model is the creation of annotations for images that can be used to train an image classifier or detector. In this section we discuss work performed in (Pham et al., 2010) that learns from a corpus of images with associated captions, a probabilistic alignment of names in the captions with faces in the image.

The "Labeled Faces in the Wild" dataset (Huang et al., 2007) is similar to the dataset discussed earlier and contains 11820 images with their captions that have been downloaded from a Yahoo! news. Typically an image contains multiple faces, the caption contains multiple names, and not every face in the image has a corresponding name in the caption and vice versa. Pham et al. (2010) developed an iterative EM procedure to align every face with it's correct name. This is a non-trivial task since faces contain a lot of variation due to facial expressions, pose and lighting conditions, and also person names are used in different ways (e.g. "George W. Bush", "President Bush" and "George Bush").

First all names in the captions are detected (using the previously described named entity recognizer) and clustered based on automatically detected coreference chains. The faces are automatically detected using a face detector and clustered based on a cosine metric on the parameters of a 3D morphable face model. An initial alignment of names and faces is learned from co-occurrence of the face clusters with the name clusters in the dataset. This estimate is iteratively updated using an EM-algorithm.

Although the visualness measure is not used (since persons receive a visualness score of 1), the experiments performed show that approaches that employ the salience measure consistently outperform approaches that assume that all persons in the caption are equally likely to appear in the image. The best system tested uses the salience measure for the names and the namedness for the faces, which estimates the probability that a face will be described in the text. This system achieved an F1-measure of 72.23% on a large test set of 10977 image-text pairs.

8.4.2 Textual retrieval of images

In this section we discuss an application of the appearance model: the retrieval of images from the world wide web. Our goal is to find the best images of a given entity (or entities) in a collection of pictures that have associated texts in the form of descriptive sentences, where every picture can depict possibly multiple persons or objects. Although most research on image retrieval is on an automatic analysis of the image (Datta et al., 2008), commercial search engines today still use the text surrounding the image as a major clue to the image's content. Since we have found that the appearance model offers a reasonable accurate representation of the image content, it is interesting to see to what extent this model can be used for *the retrieval of images.*

In section 8.4.2.1 we integrate the appearance model in a retrieval model, which we evaluate in section 8.4.2.2.

8.4.2.1 Probabilistic cross-media retrieval model

Statistical language modeling has become a successful retrieval approach (Croft and Lafferty, 2003). A textual document is viewed as a model and a textual query as a sequence of words randomly sampled from that model. Let the query be composed of one or more query word q_i, which are proper names or nouns representing a person or object. The language model for retrieval then computes the probability that the query $[q_1 ... q_m]$ is generated by image I_j

$$P(q_1, ..., q_m | I_j) = \prod_{i=1}^{m} ((1 - \lambda) P(q_i | I_j) + \lambda P(q_i | C)) \tag{8.1}$$

where C represents the collection of documents and λ is a smoothing factor between 0 and 1. The probability $P(q_i|C)$ is called the corpus model and assigns a non-zero probability to every word in the corpus proportional to the relative frequency of that term. The probability $P(q_i|I_j)$ is the probability of the image I_j generating the query term q_i. We estimate this probability from the text T_j associated to the image (e.g. the captions), and consider a number of methods. A first method (bag-of-words, BOW) estimates this probability as

$$P_{BOW}(q_i|I_j) \sim n(q_i)$$

where $n(q_i)$ is the number of occurrences of word q_j in T_j and k ranges over all words in T_j. A different method takes into account whether a word expresses an entity (or is part of a multi-word expression of an entity).

$$P(q_i|I_j) \sim \begin{cases} n(q_i) * w(e_i) & if\, q_i\, expresses\, an\, entity\, e_i \\ 0 & otherwise \end{cases}$$

Here $w(e_i)$ is a weight assigned to the entity e_i expressed by the word q_i. A first model (BOE) assumes that all entities are weighted equally, i.e. $w(e_i) = 1$. We then define a number of models that set the weight respectively to the visualness (VIS), to the salience (SAL) or to the appearance score of that entity (APP). We

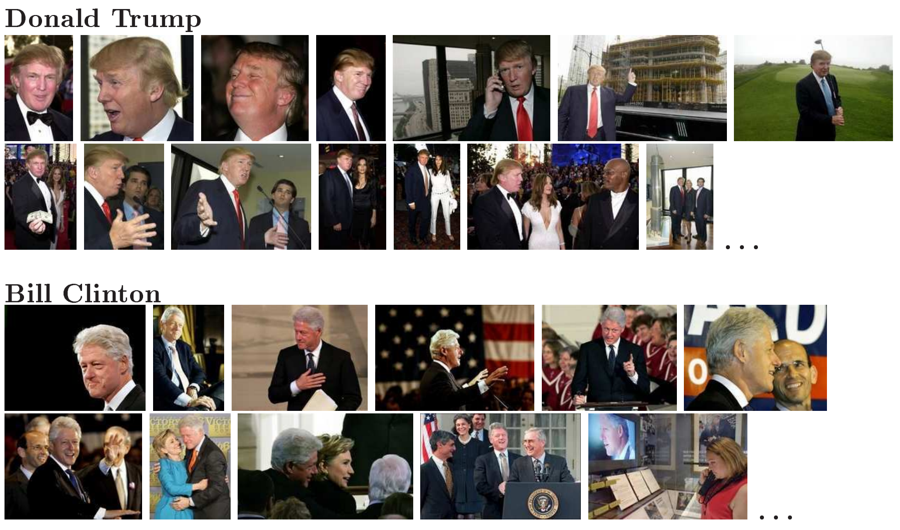


Figure 8.6: Ground truth ranking (left to right, top to bottom) for two example queries. Images are ranked higher if they contain fewer entities and if the queried entity is more prominent in the image. Images in the dataset are all approximately the same size and this factor is not taken into account for the ranking.

have adapted the Lemur toolkit[7] to include these retrieval models, using a λ equal to 0.1.

8.4.2.2 Evaluation of the appearance retrieval model

Dataset Because of the lack of a standard dataset that fits our tasks and hypotheses, we annotated our own ground truth corpus. We randomly select 700 image-text pairs from the Yahoo! news corpus discussed in section 8.1.5.1. In this selection, many images picture only a single person or object. This makes retrieval easy, since these images should be ranked at the top, and we refer to this dataset as the EASYSET. From this set we select a subset of pictures where three or more persons or objects are shown, which varying degree of prominence in the image. We call this dataset that comprises 380 image-text pairs the DIFFICULTSET. Tests on the latter set allows us to better understand the behavior of our different indexing methods when many persons or objects with varying degree of prominence are shown in the photographs.

We have randomly generated a number of queries as follows: assuming that the queries reflect the corpus used for retrieval, we select 79 images from this corpus,

[7]http://www.lemurproject.org/

	BOW	BOE	VIS	SAL	AP
EASYSET	58.12%	62.46%	55.62%	56.25%	59.28%
DIFFICULTSET	70.48%	73.46%	71.16%	69.54%	71.70%

Table 8.3: Results in terms of mean average precision for the ranking models based on the different text representations for the EASYSET and DIFFICULTSET.

and for every selected image create a query by concatenating all entities in this image. In this way we obtained 53 queries that contain one name of a person or object, and 26 queries with two entities (23 queries with two person names and 3 queries with a person and object name). For every query we ranked all images in the corpus that contained these entities according to the prominence of the entities in the images. Rankings for two examples queries can be see in figure 8.6.

Experiments For every query we rank all documents in the collection using the methods defined above, and compare the automatic rankings with the manual rankings using the mean average precision metric (table 8.3).

First, we see that the appearance measure (AP) improves the retrieval model compared to the baseline (BOW) method. This measure determines approximately how many entities in a given text are likely to appear in the image, and thus to create a more fine-grained ranking (since images with a small number of entities are preferred above images with a large number of entities). However, disappointingly this method, and all other methods are outperformed by the bag-of-entities (BOE) method, showing that the prominence is sufficiently captured by the maximum likelihood estimation of the term occurrence in the text. This is caused by a strong correlation between the length of a caption and the number of entities shown in the corresponding image. Another important factor is that the queries by definition only contain entities that can be perceived on the images, thus making an automatic analysis of the visualness of entities superfluous. Also the occasional misclassification of entities by the appearance model (see table 8.1.5), reduces the accuracy of the automatic ranking. These unexpected results show that information retrieval has different requirements then information extraction, and that an improved method for the latter does not necessary improve the former.

8.4.2.3 Related work

Image retrieval is a well studied problem with a large body of research, we refer to Smeulders et al. (2000) and Datta et al. (2008) for extensive overviews. Most research however focuses on content-based image retrieval, i.e. on methods that perform an automatic analysis of the images while only few research of using associated natural language texts has been performed.

Smeaton and Quigley (1996) and Flank et al. (1995) develop retrieval models for image captions based on respectively a WordNet distance or on finite state machines representing the syntactic structure of a sentence. These approaches should however be considered generic information retrieval models since they are not designed specifically for image retrieval. The WebSEEk search engine Chang et al. (1997) uses a manual classification of terms in associated web-pages as visual or non-visual. The visual terms are further manually mapped into different categories. The performance of this method is not reported.

The ImageClef 2009 shared task on image retrieval (Paramita et al., 2009) compares a number of systems that perform image retrieval on a corpus of almost 500.000 images with associated captions, similar to the corpus employed here. Given a query consisting of textual keywords and an example image, the systems performed a ranking of all image-text pairs in the system. All the participating systems used standard retrieval models to compare the key words with the captions, which suggests that for this type of data a more elaborate analysis of the captions does not improve retrieval performance.

The experiments performed by Xia et al. (2009) suggest that an analysis of language is necessary however when the texts associated with the images are longer and contain more words that do not describe the image. Xia et al. (2009) performed experiments where, for a given textual query, a ranking was created of images based on text extracted from the containing web page. They found that a retrieval model that uses the visualness measure outperforms a $tf * idf$ based language model by 15% to 20%.

8.5 Conclusion

In this chapter we have introduced the appearance model, consisting of an analysis of the text combined with external knowledge. We have designed a method to capture the salience of the entities, based on analysis of the discourse of the text and of the syntactic structure of the sentences. We have combined this with a new model of the visualness of the entities, that employs a distance metric defined on WordNet together with a small number of seed synsets. We have shown that this method can predict which entities and attributes are present in an image, without performing an analysis of the image itself. We have then described two applications of this model: the alignment of names and faces and image retrieval. It was shown in these experiments that the salience model helps to accurately align names and faces by indicating which persons are more likely to appear in the image. We then saw that, although existing retrieval models are sufficient for short captions, the visualness model did help for text pages that contained longer texts with more entities not visible in the image.

Chapter 9

Automatic annotation of video

In the previous chapter we have discussed the difficulties faced by automatic methods for the analysis of images, and how automatically generated annotations from associated texts can improve these methods by providing a weak labeling. The analysis of video data however provides even greater difficulties for automatic methods. In this chapter we discuss methods that automatically generate annotations for video data using associated texts or transcripts. We focus on two types of annotations: visual actions and their semantic roles (section 9.1) and locations of scenes (section 9.2). These annotations can then be used to help methods that perform an automatic analysis of the video. We conclude this chapter in section 9.3.

9.1 Visual action annotation

In this section we investigate methods for the automatic annotation of actions in video. To this end we apply the previously developed semantic role labeling system to semantic roles of *visual verbs*. We introduce this task in section 9.1.1 and describe the role definitions in section 9.1.2. We evaluate the dataset used in this application in section 9.1.3 and evaluate it in section 9.1.4. Finally we describe its use in automatic annotation in section 9.1.5.

Movement	A person moving voluntarily from one position to another ("Buffy **walks** in the room").
Object manipulation	A person manipulating or moving an object (".. Buffy **opening** the refrigerator").
Body position	A verb describing the position or pose of the body of a person (".. Buffy **leans** forward..").
State	A verb that describing the stationary state of an object or person, different from the body position ("... Dawn **is** chained to the wall.").
Express emotion	A person expressing some emotion ("Dawn **shrugs** in embarrassment").
Fighting	Actions performed by a person in a fight ("... Harmony **backhands** Anya").
Camera action	The camera zooming in or out or moving with respect to the scene ("The camera **pans** across a bedroom").
Visual	Any visual verb that does not belong to one of the above categories (".. Buffy **talking** to Giles").

Table 9.1: Semantic frames for automatic video annotation.

9.1.1 Introduction

Action detection and classification in video is a hard task that has only recently been approached outside laboratory conditions, and is typically still limited to a small number of actions (Laptev et al., 2008). Furthermore the manual segmentation and labeling of actions in video is a labour-intensive and error-prone task. Following our work on automatic image annotation in the previous chapter, we would like to develop methods for the automatic annotation of actions and their arguments (i.e. semantic roles) in videos. We are hereby only interested in *visual* actions and arguments, i.e. actions and arguments that can be perceived in a single frame or in a sequence of video frames. This work will show that we can easily adapt our semantic role classifier to different sets of semantic roles and frames.

9.1.2 Semantic roles

The PropBank semantic roles described in section 4 define a set of semantic roles and a set of senses for every verb separately. This definition is however not very useful for automatic image annotation since typically it is required to generalize across different verbs. We thus define a new set of semantic frames that are motivated by the requirements of automatic image analysis: all verbs labeled with an identical frame should have a more or less similar *visual appearance* in the

Agent The person or object performing the action ("**Buffy** walks in the room").

Causative agent The person or object that force another person or object into the action ("**Buffy** pushes her back").

Patient The person or object forced into the action ("Buffy pushes **her** back").

From position The initial position before movement ("Buffy picks up a banana **from a bowl of fruit**").

To position The final position after movement ("she crashes **to the floor**").

Emotion The emotion that is expressed ("Buffy's face looks very **peaceful**").

Negation The word indicating negation of a certain action ("Willow does **not** notice her expression").

Table 9.2: Semantic roles for automatic video annotation.

video, and vice versa. Furthermore the definitions should reflect the actions that occur frequently in the dataset used and should include actions of the camera (e.g. "zoom"), which is potentially useful for visual action detection methods. In dialog with researchers more experienced in video analysis and taking into account the dataset, we have created a new set of frames (listed in table 9.1) and semantic roles (table 9.2).

9.1.3 Dataset

We apply the proposed semantic frames and roles to the transcripts of a popular action series, *Buffy, the Vampire Slayer*. This American TV series stars Buffy Summers and her friends as they fight vampires and other demons. It offers an interesting testbed for our automatic annotation techniques: because of its large popularity, fans have created transcripts for all episodes that offer detailed descriptions of the video and the dialog. These transcripts contain information on the actions of the characters, their emotions and of the locations of the scenes. This information is however all embedded in running text and need to be extracted with information extraction techniques. Also the video offers a realistic testbed for video analysis because of the challenging lightning conditions, frequent motion blurring and variations in pose and camera position.

A human annotator has manually annotated all verbs and their semantic roles in episodes 1 to 9 of the fifth season, totaling 4340 frames with 12754 roles. These annotations have been checked for inconsistencies by a second annotator.

model	%P	%R	%F1
generative	70.36	63.61	66.79
discriminative	**76.77**	**74.75**	**75.85**

Table 9.3: Results for semantic role detection for visual verbs.

Shot of Buffy opening the refrigerator and taking out a carton of milk. Buffy sniffs the milk and puts it on the counter. In the background we see Dawn opening a cabinet to get out a box of cereal. Buffy turns away.

Obj. mani.	**opening**
Agent	Buffy
Patient	the refrigerator

Obj. man.	**taking out**
Agent	Buffy
Patient	a carton of milk

Obj. man.	**opening**
Agent	Dawn
Patient	a cabinet

Movement	**turns away**
Agent	Buffy

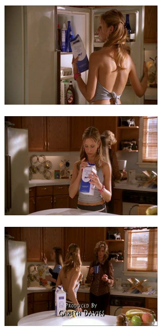


Figure 9.1: SRL result (middle column) for an example scene, showing the transcript (left column) and video frames (right column).

9.1.4 Evaluation

The transcripts were preprocessed by removing all HTML formatting. We have also removed all dialog, since this contains very little information on the actions of the characters. The descriptive text was then split in sentences that were part-of-speech tagged (Mikheev, 1997) and parsed (Charniak, 1997). We created features from these tags (see section 4.3.1), trained our generative and discriminative semantic role classifiers (see section 4) on 8 episodes and tested on the remaining episode. Table 9.3 shows that also here the discriminative model outperforms the generative model, confirming our results in the previous chapters. We also show some examples of automatically detected semantic frames in figure 9.1. We see how they accurately describe the actions in the video.

We would like to emphasis again that the SLR systems and features applied here are exactly the same as used on the PropBank dataset, showing that our approach is very flexible and can be applied rapidly on new datasets.

9.1.5 Automatic image annotation

We have described semantic role detection in text for visual actions. Given a text describing an image or video, we can use the developed system to automatically generate annotations that can be used in image analysis. Jie et al. (2009) have used our system to successfully detect the actors of action verbs in texts describing news images. From these automatic annotations they learn associations between names and visual face descriptions and between action verbs and visual pose estimates. The learned associations using the automatically detected actors are appr. 75% correct for both name-face and action-pose associations, which compares favorably to associations learned with manually annotated actors, resulting in appr. 80% accuracy for name-face and appr. 83% accuracy for action-pose associations.

In the future we would like to use the annotations generated by this SRL classifier for analyzing other actions, such as emotions or movements.

9.2 Scene location annotation

The work described in this section is joint work with Chris Engels, Jan Hendrik Becker, Tinne Tuytelaars, Marie-Francine Moens and Luc Van Gool

In this section we consider the problem of annotating scenes in a video with information extracted from an associated text. We introduce this problem in section 9.2.1 and describe how we use a multimodal approach to scene segmentation (section 9.2.2) and scene annotation (section 9.2.3). We evaluate our approach in section 9.2.4.

9.2.1 Introduction

We consider a video (e.g. a motion picture or soap series) that has an associated text (e.g. a transcript) that describes the content of the video. From this text, we aim to extract the location of a particular scene in the video. These annotations could be presented to an end-user, used for a textual search in a video-archive, or as a weak annotation for visual scene classifiers. An important difficulty is that the number of locations is not known beforehand and that many locations will only occur in a single video. We thus need a method that is able to dynamically determine the number of and textual descriptions for locations in a new video. We hereby rely on information extraction methods that extract the location descriptions from the associated texts. Furthermore we develop a method to propagate location annotations from one scene to scenes that are visually similar.

In this work we use transcripts for an action series that is created by fans (see section 9.2.4). These transcripts contain descriptive text together with the dialog. Figure 9.2 gives an overview of our approach. We begin by roughly aligning the transcript to the video using subtitles with approximate timing information. We then split the video into scenes using a text classifier and shot cut detector. For every scene we extract the location phrases and use these to train a latent Dirichlet

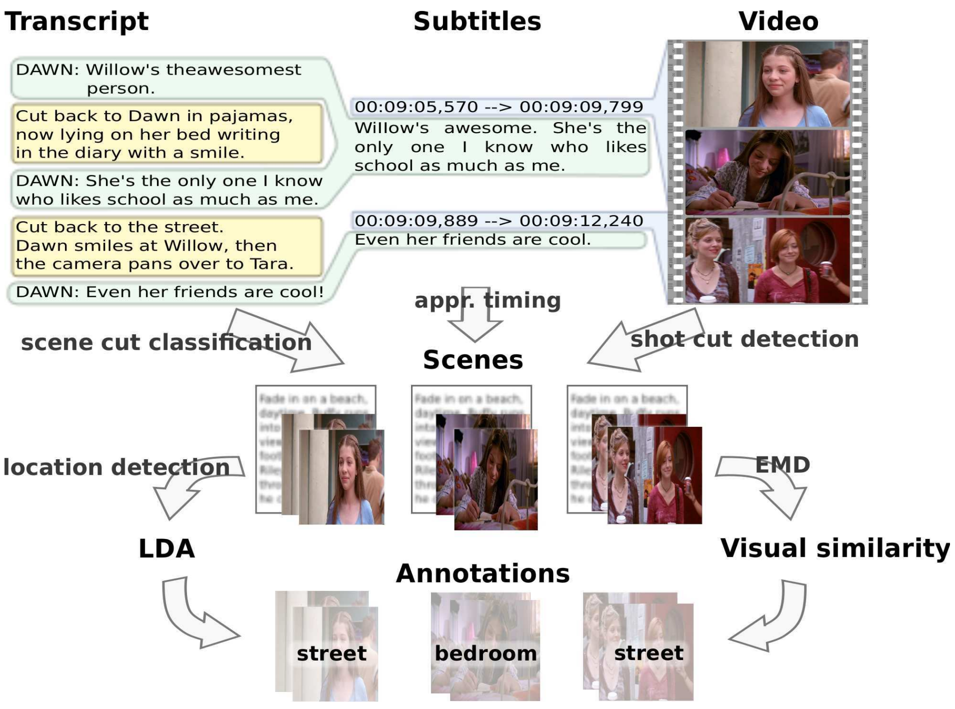

Transcript
Subtitles
Video
DAWN: Willow's theawesomest person.
Cut back to Dawn in pajamas, now lying on her bed writing in the diary with a smile.
DAWN: She's the only one I know who likes school as much as me.
Cut back to the street. Dawn smiles at Willow, then the camera pans over to Tara.
DAWN: Even her friends are cool!
00:09:05,570 --> 00:09:09,799
Willow's awesome. She's the only one I know who likes school as much as me.
00:09:09,889 --> 00:09:12,240
Even her friends are cool.
appr. timing
scene cut classification
shot cut detection
Scenes
location detection
EMD
LDA
Visual similarity
Annotations
street
bedroom
street

Figure 9.2: Overview of our approach

allocation topic model. The final annotations are then chosen as the phrase that is most likely given the topic distribution, reweighted by visual similarity.

9.2.2 Scene Segmentation and Alignment

We define a scene as a consecutive sequence of shots that are set in the same location. Scenes are used as the basic units that are annotated in our work. To segment the video into scenes we use a multimodal approach, combining a shot cut detector with a text classifier.

9.2.2.1 Coarse Alignment

We first create an approximate alignment of video and text. The transcripts used here do not contain any timing information. To obtain approximate timing information, we align the dialog in the transcript with the subtitles extracted from the video, using the time-warping approach described by Everingham et al.

(2006) (Sankar K. et al. (2009) used speech recognition for the same purpose). The timing information from the subtitles can thus be transferred to the dialog in the transcripts, which also gives an approximate timing of the descriptions, since these are interwoven. This timing is however only approximate, and becomes worse in scenes with limited speech. We further refine this alignment in section 9.2.2.4.

9.2.2.2 Learning scene cuts in the text

The descriptive part of the transcripts often contains strong cues for the start of new scenes, e.g. "Fade in on a beach, daytime.". The dialog part does not contains these clues, and we thus discard the spoken lines. We learn a sentence classifier that classifies every sentence as describing a transition from one scene to another (e.g. "Cut to the kitchen").

We preprocess the text by dividing the textual descriptions into sentences, tokenize the sentences into words and perform part-of-speech tagging (using the LTPOS tagger (Mikheev, 1997)). We then extract the following features

Unigram Every word token in the sentence (e.g. "buffy").

Bigram Every consecutive sequence of 2 tokens in the sentence (e.g. "buffy_running").

Trigram Every consecutive sequence of 3 tokens in the sentence (e.g. "buffy_running_throug

POS Part-of-speech tags of all words in the sentence (e.g. "VB", "NP").

POS+token Part-of-speech tag concatenated with the word token (e.g. "NNP_buffy").

Position The position of the sentence in the text, given by the character pointer. This position is binned in 20 intervals of equal width.

We perform experiments with a generative and discriminative classifier, and find the optimal combination of features for every classifier. We use the classifier to compute for every sentence $\mathbf{w} = \{w_1, \ldots, w_n\}$ of n words w a probability $P_{\text{cut}}(\mathbf{w})$ for the occurrence of a scene cut in this sentence.

9.2.2.3 Detecting visual shot cuts

We localize our scene cuts by detecting an associated shot cut in the video. Shot boundary detection is fairly well-established, see e.g. Yuan et al. (2007) for a comprehensive review. Our implementation uses a sliding window over color histograms to compute a dissimilarity energy E_{cut} based on χ^2-distance, followed by local non-maximum suppression and thresholding.

9.2.2.4 Refined scene cuts

As mentioned before, the alignment for text descriptions may be imprecise in scenes with little dialog. To minimize this error, we need to refine the alignment of sentences near a scene cut boundary.

We first quantify the error of our initial textual and visual cuts relative to ground truth using our training episodes. We assume the distribution of the timing errors in both text and visual cuts is Gaussian, and we learn the mean offsets $\mu_{\text{text}}, \mu_{\text{vid}}$ and standard deviations $\sigma_{\text{text}}, \sigma_{\text{vid}}$ of the text and video, respectively. To determine the exact frame of the scene cut, we define energy terms for each modality. For a frame k, the text cut energy E_{text} is

$$E_{\text{text}}(k) = \max_i P_{\text{cut}}(\mathbf{w}_i)\mathcal{N}(t_k|(t_i + \mu_{\text{text}}), \sigma_{\text{text}}) \tag{9.1}$$

where $P(\mathbf{w}_i)$ is the probability of a cut occurring at sentence $\mathbf{w}_i$ and $\mathcal{N}(x|\mu, \sigma)$ is a Gaussian distribution evaluated at x. t_i is the initial time estimate of sentence $\mathbf{w}_i$, which is chosen as the end time of the subtitle occurring just before this sentence. Similarly, we define the video cut energy E_{vid} as

$$E_{\text{vid}}(k) = \max_i E_{\text{cut}}(C_i)\mathcal{N}(t_k|(t_i + \mu_{\text{vid}}), \sigma_{\text{vid}}) \tag{9.2}$$

for a detected cut C_i at time t_i.

Our final cuts are found by performing local non-maximum suppression and thresholding on the joint energy

$$E_{\text{TV}}(k) = E_{\text{text}}(k)E_{\text{vid}}(k) \tag{9.3}$$

Figure 9.3 shows an example of the energies evaluated over some frames, where a red `X` denotes a ground truth cut.

9.2.3 Location annotation

For every automatically detected scene we want to generate a text phrase that describes the location of that scene. Here for we use a topic model that uses extracted locations from the text together with a visual similarity of the video scenes. We start by describing the location classifier.

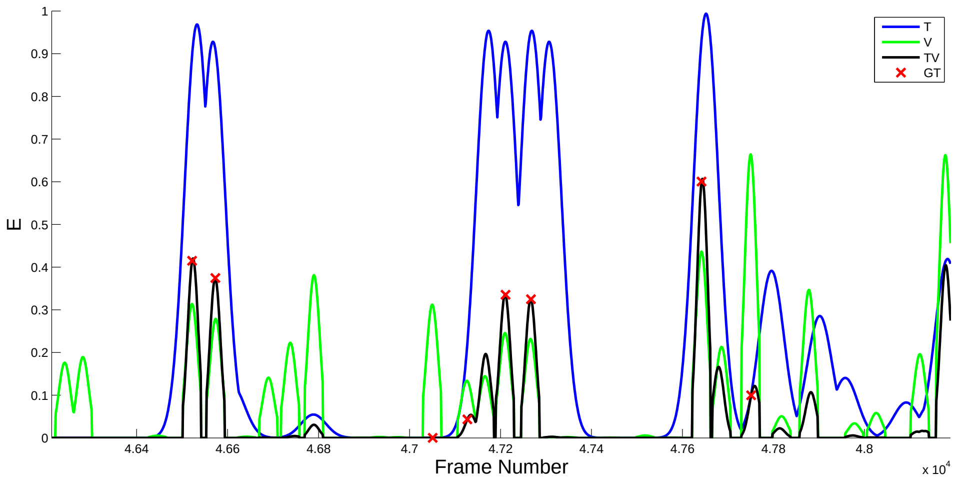


Figure 9.3: Example of cut energy over time. The blue line corresponds to E_{text}, green to E_{vid}, and black to E_{TV}. A red X denotes a ground truth cut.

9.2.3.1 Identifying location phrases

We want to determine for all phrases in the text, whether they describe a location or not. We manually annotate the locations in a number of transcripts, and assign a label to all words: `bLoc`, `iLoc`, `oLoc` for respectively the first word in a phrase describing a location, other words in this phrase or words outside these phrases. We want to compare a generative and a discriminative sequential classifier and train a hidden Markov model or maximum entropy Markov model (Ratnaparkhi, 1996) on manually annotated episodes and use these to label unseen episodes. For every word we generate the following features:

Token The word token (e.g. "bedroom").

POS Part-of-speech tag of the word (e.g. "DT").

Scene cut probability The binned probability of the scene cut classifier for that sentence.

Possessive Boolean value indicating whether this word is in a possessive form (e.g. "Joyce's").

Previous token The word token that occurs just before this word.

Next token The token that occurs just after this word.

Path to top The path from the root node of the parse tree to this word (e.g. root↓dep↓nmod↓pmod).

Hidden word A probabilistic feature representing the hidden word distribution as determined by the latent words language model.

Although not listed here, the hidden Markov and maximum entropy Markov models also take into account the label of the previous word. We use these classifiers to compute a probability $P_{loc}(\mathbf{w}_{i-n+1}^{i})$ for every phrase $\mathbf{w}_{i-n+1}^{i}$.

9.2.3.2 Latent Dirichlet allocation

Many scenes do not mention the location explicitly, although the location could be inferred by the description of other objects (e.g. "fridge" in the kitchen). Furthermore, the textual descriptions of locations contain significant variation, posing various problems for learning location labels for scenes. Problems arise from synonyms and polysemes, where multiple phrases are used to refer to the same location (e.g. "cemetery" and "graveyard"), or where different locations are referred to by the same phrase (e.g. "the living room" refers to the living room in two different houses).

We use latent Dirichlet allocation (LDA, Blei et al. (2003)) to address these problems. LDA learns, from a corpus of documents, probabilistic topics that capture soft clusters of words that occur frequently together. It is a generative model of documents where the generative process is summarized as follows: for a document d a multinomial mixture parameter θ is first sampled. Then, for each word w a topic z is sampled from the multinomial distribution and, the word w is sampled from the multinomial word distribution conditioned on that topic. The probability of a collection D of M documents is given by

$$P(D|\alpha,\beta) = \prod_{d=1}^{M} \int P(\theta_d|\alpha) \left(\prod_{i=1}^{N_d} \sum_{z_{di}} P(z_{di}|\theta_d) P(t_{di}|z_{di},\beta) \right) d\theta_d$$

where θ_d is the topic distribution for a document d, t_{di} is the term on position i in document d and z_{di} is the topic assigned to this term. LDA is trained on this corpus by finding the parameters α and β that maximize the likelihood of the model on the data. This model alleviates the aforementioned problems by assigning synonyms to the same topic and assigning polysemous words to multiple topics. Context words contribute to topics as well and as such can help to identify the location in case of underspecification. In our work the documents correspond to scenes, and we set the terms of a scene to all the extracted phrases from the text of that scene.

To make sure that the learned topics reflect the different locations and not other topics in the text, we assign a weight $v(\mathbf{w}_{i-n+1}^{i})$ to every sequence of words $\mathbf{w}_{i-n+1}^{i}$

of length n. These weights reduce the influence of phrases that are not indicative of location and force the topics to focus on location information. The parameters α and β are then chosen to optimize

$$P(D|\alpha,\beta) = \prod_{d=1}^{M} \int P(\theta_d|\alpha) \left(\prod_{i=1}^{N_d} \sum_{z_{di}} P(z_{di}|\theta_d) P(\mathbf{w}_{i-n+1}^{i}|z_{di},\beta)^{v(\mathbf{w}_{i-n+1}^{i})} \right) d\theta_d$$

We set the weight of a phrase to the probability of that phrase describing a location, i.e. $v(\mathbf{w}_{i-n+1}^{i}) = P_{loc}(\mathbf{w}_{i-n+1}^{i})$, as given by the location classifier described in the previous section.

9.2.3.3 Visual similarity

Some scenes lack text that describes the location (nor describes other informative objects), so their respective topic distributions will not be useful in generating an annotation. In these cases, we can use visually similar scenes to propagate critical words to the ambiguous scenes.

Given two scenes, we need to compute a measure for the visual similarity of the scene locations. In the foreground of a typical scene, there are often one or more persons present. The background may be cluttered, out of focus, sparsely detailed, and occluded by people. Additionally, the camera perspective may be stationary, move smoothly, or frequently cut away, potentially causing the background to appear completely different from alternate viewpoints.

Persons themselves are not indicative of a certain location, as they may appear in different locations. Therefore, we use the upper-body pose detector of Ferrari et al. (2008) to excise them from the scenes as much as possible, prior to computing a visual scene descriptor for the scene similarity measure.

Visual description Several methods exist for the description of the visual appearance of a general scene. Local or GIST features (Oliva and Torralba, 2006) are not suitable for our purpose, as we need a descriptor that does not encode the scene in too much detail, but is rather robust against e.g. backgrounds being in or out of focus. Therefore we convert the colors in the background of all videos to the CIELab color space (Wyszecki and Styles, 1982) and cluster them in 32 clusters using k-means clustering. Every frame is then described by the ratio of its background colors in every cluster.

Next, we cluster shots in each scene into a small number of distinct camera perspectives, and store the mean histogram for each cluster along with the corresponding number of images. We use the self-tuning spectral clustering method proposed by Zelnik-Manor and Perona (2004), which handles multiple scales of clusters and provides a convenient way of selecting the optimal number of clusters.

Visual distance of scenes using EMD To estimate the distance between two scenes s and s' from the set S of all scenes we use a nested Earth Mover's Distance (EMD) approach (Rubner et al., 2000). EMD measures the distance between two distributions of weighted clusters $\{(c_1, \omega_1), (c_2, \omega_2), \ldots, (c_n, \omega_n)\}$ by solving a flow optimization problem using pairwise costs between clusters. There is no requirement for each distribution to have the same number of clusters.

In our work shots are represented by weighted color clusters and scenes are represented as weighted shot clusters. EMD is first used to compute a pair-wise distance between shots, and these distances are used in a second iteration of EMD to compute a distance $\text{EMD}(s, s')$ between scenes s and s'.

Finally, we convert these scene distances into a similarity matrix:

$$A(s, s') = \exp\left(-\frac{1}{\lambda}\,\text{EMD}(s, s')^2\right) \tag{9.4}$$

where λ is a scaling parameter determined from the training data.

9.2.3.4 Updating the topic distributions

For scenes that do not contain enough words indicative of location, the topic distributions obtained solely from the transcript are inadequate, despite the reweighting. Therefore, inspired by the Mixture of Experts model (Jacobs et al., 1991), we model an updated topic distribution $\tilde{P}(z_i|s)$ as a mixture of the original topic distributions:

$$\tilde{P}(z_i|s) = \sum_{s' \in S} \pi(s, s') P(z_i|s) \tag{9.5}$$

The mixing coefficients $\pi(s, s')$ are given by the normalized visual similarity $A(s, s')$ between scenes s and s':

$$\pi(s, s') = \frac{A(s, s')}{\sum_{s' \in S} A(s, s')} \tag{9.6}$$

This effectively allows to propagate location labels between visually similar scenes.

9.2.4 Experiments and evaluation

We evaluate our system on 4 episodes of *Buffy the Vampire Slayer*, for which transcripts are readily available on the Internet (Twiz TV). These episodes

combination	%P	%R	%F1
best = bigram + W&P + position	**91.71**	79.48	**85.16**
best + unigram	90.36	76.92	83.10
best - bigram	88.41	74.35	80.77
best + trigram	91.62	78.46	84.53
best + POS	90.17	80.00	84.78
best - W&P	88.70	**80.51**	84.40
best - position	90.47	77.94	83.74

Table 9.4: Accuracy of the discriminative scene cut classifier for different combinations of features. + adds a feature, − removes a feature, *POS* is part-of-speech tag, *W&P* is the concatenated word token and part-of-speech tag and *position* is the binned position of the sentence in the transcript.

provide a challenging validation for our system since the textual transcripts are unstructured and contain a lot of variation, and the video has highly variable lighting conditions, frequent motion blurring and many different locations. We evaluate our system on episodes 1 to 4 of season 5. On average an episode has appr. 53 scenes in appr. 20 different locations. Only a handful of these locations are shared across episodes.

We evaluate the different parts of our system: scene cut detection (section 9.2.4.1), location detection (section 9.2.4.2), and the automatic annotations (section 9.2.4.3).

9.2.4.1 Scene cut evaluation

We first perform an evaluation of the generative and discriminative scene cut classifiers. We have manually annotated the scene cuts in the transcripts of 4 episodes and perform 4-fold cross validation, training the classifier on 3 episodes and testing on the remaining episode. For every classifier we find the optimal combination of features (table 9.4), which was *bigram+W&P+position* for the discriminative classifier and *bigram+W&P* for the generative classifier. Table 9.5 shows that the performance of the generative classifier is close to the performance of the discriminative classifier. This differs from results in the previous chapters, where discriminative classifiers significantly outperformed generative classifiers for word sense disambiguation and semantic role labeling. Here however we face a very small training set (3 episodes) on which generative classifiers will often perform quite well, since they have a smaller variation then discriminative classifiers when trained on a limited number of samples (Bouchard and Triggs, 2004).

The lower recall of the scene cut detectors is mainly caused by incorrect

classifier	%P	%R	%F1
discriminative	**91.71**	**79.48**	**85.16**
generative	91.07	78.46	84.29

Table 9.5: Performance of the discriminative and generative scene cut classifiers in terms of precision (P), recall (R) and F1-measure.

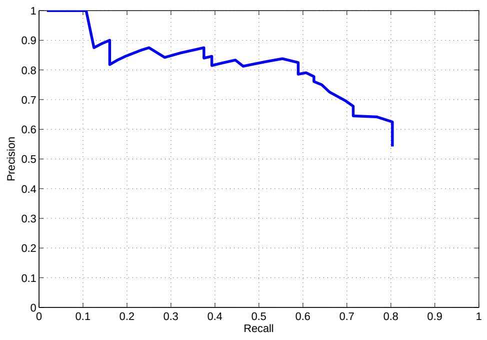


Figure 9.4: Precision-recall of final scene cuts with fixed error margin of 12 frames.

classification of sentences that described the actors moving from one location to another, e.g. “Buffy goes into another room...”, which are also considered scene cuts because of the location change.

To find the combined scene cuts in video and text, we combine the probabilities generated by the text classifier with the detected shot cuts in the video as described in section 9.2.2.4. The resulting confidence values give the precision-recall curve in figure 9.4. Many of the missed scene cuts are in areas with either few text descriptions or dialog, leading to low cut probabilities or imprecise cuts, respectively. We select a cut threshold with a high recall, since the visual update step provides robustness to oversegmentation.

9.2.4.2 Location detection evaluation

In this section we evaluate location detection in text. As described in section 9.2.3.1 we perform experiments with two classifiers, a generative hidden Markov model and a discriminative maximum entropy Markov model. For both models we perform 4-fold cross validation on 4 manually annotated episodes. We first find the optimal combination of features (table 9.6), which was *token+cut_prob+previous word + next word+hidden word* for the HMM and *token + POS+ cut_prob + possessive + previous word + next word+hidden word* for the MEMM. We see in

combination	%P	%R	%F1
best=token + cut_prob + previous + next word	68.75	75.54	**71.98**
best - token	50.90	36.05	42.21
best + POS	56.92	79.39	66.30
best - cut_prob	59.41	75.31	66.42
best - previous word	56.53	**78.48**	65.72
best - next word	61.53	57.87	59.64
best + possessive form	**70.66**	73.07	71.84
best - hidden word	69.72	70.05	69.88

Table 9.6: Accuracy of the discriminative MEMM location detector for different combinations of features. + stands for adding a features, − for removing, *POS* for part-of-speech tag and *cut_prob* for the binned scene cut probability of the sentence.

classifier	%P	%R	%F1
HMM	68.75	**75.54**	**71.98**
MEMM	**81.31**	61.92	70.30

Table 9.7: Performance of the generative HMM and discriminative MEMM location detectors in terms of precision (P), recall (R) and F1-measure.

table 9.7 that the HMM outperforms the MEMM. This again shows that generative models can outperform discriminative models when trained on small training sets. The most common error by both classifiers is incorrect segmentation, where only part of a location is correctly labeled (e.g. labeling “the house” instead of “inside the house”). This however typically results in labels that are still informative of the location to the end-user.

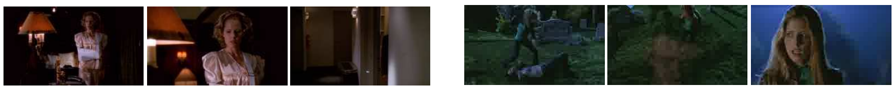
9.2.4.3 Location annotation evaluation

We evaluate the correctness of the annotations generated by our system. We could evaluate the number of scenes that have a correct annotation, but this would depend on the number of scenes that are automatically detected and would treat long scenes equal to short scenes. Instead we transfer the annotation of a scene to every frame in that scene and manually count the number of frames in the video that have been assigned a correct annotation.

Table 9.8 shows the performance of our system on two episodes (episode 2 and 3 from the 5th season). We see that for episode 2 the LDA topic model reduces the errors by 9.4%. For this episode the visual reweighting did not help. A

episode	only text	text + LDA	text + LDA + vision
2	54.72%	**58.98%**	57.39%
3	60.11%	65.87%	**68.67%**

Table 9.8: Performance of automatically generated annotation as judged by a human annotator, excluding opening and closing credits.



Transcript	Cut back to Anya's. Anya looks conflicted.	**Transcript**	Various shots of Buffy fighting and killing vampires
Text only	anya's	**Text only**	various
Combined	anya's apartment	**Combined**	a graveyard

Figure 9.5: Two scenes with a selection of their video frames, description in the transcript and the automatic annotations based on the text only or on the combined text and video.

manual inspection revealed that a number of incorrect scene cuts confused the visual similarity metric by merging a number of scenes in one large segment.

For episode 3 we see that also here the LDA topic model reduces errors with 14.4%. For this episode the visual reweighting improves the automatic annotation, resulting in a total error reduction of 21.46%. We show some example scenes in figure 9.5. These scenes illustrate that annotations are successfully transferred from visually similar scenes when they are lacking in the textual description.

9.2.5 Related Work

Locations are of interest to and well-explored by several branches within the computer vision and robotics communities. Generic scene type classification, which seeks to describe the kind of location seen in an image (e.g. beach or street) has been studied e.g. in (Vogel and Schiele, 2007; Lazebnik et al., 2006; Vailaya et al., 2001; Blighe and O'Connor, 2008; Ni et al., 2008). Such approaches mostly rely on supervised techniques and large sets of annotated training data.

Recently, some weakly supervised methods for automatic video annotation have been proposed. Some of these methods focus on detecting similar locations, such as Schaffalitzky and Zisserman (2003), who develop a model that retrieves images of a particular location based on wide baseline matching techniques, Vailaya et al. (2001) who clusters images in coarse categories such as city/landscape, forest/mountain, Schroff et al. (2009) represent frames with texton histograms and cluster these in a number of locations with single-link agglomorative clustering, and

Héritier et al. (2007) who use latent aspect models to identify discriminative and often reoccurring parts of locations using SIFT features, which are then labeled manually. These methods only create clusters of shots in the same location and do not attempt to assign a label or textual description to these clusters.

Other researchers focus on scene segmentation e.g. Zhai and Shah (2005) use a purely visual Markov chain Monte Carlo approach and Chen et al. (2008) use a time-constrained clustering algorithm. These methods do not attempt to classify the obtained scenes in any way. Zhu and Liu (2009) study the problem of segmentation into scenes, and classify the obtained scenes into either conversation, suspense, or action scenes, based on audio and video and using heuristic rules for the actual classification. Neither of these works explores the use of fan scripts to obtain location annotations automatically.

Other authors have looked into the use of readily available textual annotation for TV and movie footage to learn to annotate in a weakly supervised manner as well. In particular, Cour et al. (2008) propose a unified generative model that integrates scene segmentation, script alignment, and shot threading. Everingham et al. (2006) use fan scripts aligned to the video data based on the subtitles to then identify the cast in a soap series. Laptev et al. exploit scripts for action recognition in Hollywood movies, using a supervised text classifier (Laptev et al., 2008) and using a kernel-based discriminative clustering algorithm to overcome problems with inaccurate alignment between video and text (Duchenne et al., 2009). Finally Marszałek et al. rank video segments based on actions, using mining techniques (Marszałek et al., 2009). They also mine location names, but using scripts that are way more structured than ours and not focusing on specific locations but rather scene types.

In our work, we use a purely textual topic model (i.e. LDA (Blei et al., 2003)). Other people have investigated the use of cross-modal topic models, combining visual and textual information, e.g. (Blei and Jordan, 2003; Monay and Gatica-Perez, 2003; Li et al., 2009), in the context of automatic image annotation. However, it turns out it is relatively difficult to balance the contributions of both modalities. Moreover, in our application, text and visual information are only weakly linked, with often complementary information present in only one of the two modalities. Hence, we decided to use the visual information in a postprocessing step to the textual topic model, updating the textual topic distributions based on visual similarity. This is, in some sense, similar to the tag propagation proposed by Guillaumin et al. (2009).

9.3 Conclusions of this chapter

In this chapter we have developed information extraction methods that analyze textual descriptions of a video. In the first section we developed a SRL classifier for visual verbs. We have shown that we can apply the model developed in earlier chapters to new frame and role definitions and to new datasets. This classifier was successfully used to learn associations between descriptions and images for persons and their pose.

In the second section we developed a novel multimodal approach to weakly supervised automatic annotation of locations from video and text. We have first described how scene cuts are detected by combining a scene cut detector in the text with a shot cut detector in the video. The combination took into account the approximate alignment of the two media. We have then developed a novel method for the detection of locations in the text and combined this model with visually reweighted LDA, which allowed the propagation of locations to visually similar scenes. This system was tested on a challenging action series with many infrequent locations, where the transcripts often do not describe the locations. The evaluation showed that in many cases we were able to detect a location present in the descriptions, and that if the text was lacking a description we could propagate an annotation from a similar scene, where both LDA and the visual similarity had a positive contribution, ignoring the case where the visual similarity was confused by a incorrect scene segmentation.

In the future we would like to exploit a combination of LDA and the latent words language model. Where LDA is very good at learning global topics for a text, and learning words that share the same general topics, the latent words language model is very good in learning synonyms and similar words in a certain specific context. A combination of these methods could result in even more precise word similarities, that take into account both local (the context) and global (the entire document) information. In this respect we mention the work by Griffiths et al. (2005) who learns simultaneously syntactic classes and semantic LDA-style topics. However in this work the words belong to either a syntactic class *or* to a topic, while we are interested in a method that would combine these in a joint model.

Part IV

Epilogue

Chapter 10

Conclusions

Summary and contributions

In this thesis we performed research on a number of topics related to information extraction from texts. Our main interest was the study of weakly supervised methods that use knowledge learned from unlabeled data to improve the performance of supervised models. We first discussed this in a uni-modal setting, where a set of labeled text is augmented with a large body of unlabeled texts to improve the accuracy of information extraction methods on texts, and then discussed this in a multimodal setting, where information extracted from descriptive texts was used to improve the accuracy of image analysis methods.

Our text started by outlining the context of our research in chapter 1, which was expanded in chapter 2 with a number of examples of popular information extraction tasks and with an extensive introduction to directed Bayesian networks.

In **part I** we applied today's standard methods for supervised information extraction to two tasks. Chapter 3 described a supervised method to word sense disambiguation and chapter 4 described a supervised method to semantic role labeling. Both methods used well-studied features and models, and achieved state-of-the-art results. These results where however not fully satisfying, especially taking into account the large amount of labour needed to create the large manually annotated datasets used for training. We argued that these supervised models suffered from ambiguity and underspecification characteristic to natural language.

In **part II** we addressed these problems with uni-modal weakly supervised learning, where a set of annotated texts was augmented with a large body of unlabeled texts. In chapter 5 we proposed a novel model for semi-supervised learning for semantic role labeling. We used hidden variables in the Bayesian models to represent the labels of the unlabeled examples and estimated the values of these variables with Gibbs sampling (for the generative model) or Metropolis-Hastings sampling (for the discriminative model). We observed however how the performance of these models deteriorated when using more unlabeled examples. We have then proposed a generative multiple-mixtures model where semantic roles were modeled with a number of mixture components that gave the model more expressive power to model natural language. This model was more robust to large numbers of unlabeled data, but did not outperform the supervised model.

We turned to a different approach in chapters 6 and 7. In chapter 6 we proposed the latent words language model. This model is a novel model of natural language that learns word similarities to reduce the sparseness problems associated with traditional n-gram models. Because of the large number of hidden variables, traditional methods for training and inference (i.e. the Baum-Welch algorithm) are not tractable for this model. We have developed a new method for inference, termed the forward-forward beam search. This method was also used when training this model and when using it to predict the probability of unseen texts. Experiments showed that this model outperformed both standard n-gram smoothing models and class-based language models. The automatic learned word similarities in this model were used in chapter 7 we improve the models for word sense disambiguation and semantic role labeling. We discussed various methods to incorporate this knowledge in the supervised models and found that an approach where the hidden words were used as probabilistic features resulted in an improvement over supervised models without these features. These improvements were largest with small training sets, showing that this method reduces the dependency of the supervised models on large annotated datasets. This method had the additional advantage that it can be easily incorporated in other supervised information extraction and natural language processing methods.

In **part III** we turned to the problem of multimodal weakly supervised learning. In this part we discussed various methods to analyze text describing the content of an image or video. We started in chapter 8 by describing the appearance model. This model combined the salience and the visualness measure to select with high accuracy the entities from the text that are likely to occur in the image. To compute the salience (i.e. the importance of an entity in the text) we combined an analysis of the discourse of the entire text with a syntactic analysis of the individual sentences. The visualness measure (i.e. the extent to which an the entity can be perceived visually) was computed with a novel method that uses the WordNet hierarchy together with a number of visual and non-visual seed synsets. We used this model in two applications, to align names in the text with faces in the image and to perform a textual image search. This model was also extended to include visual attributes, where the visualness of these attributes was learned from a corpus with image descriptions.

Chapter 9 discussed the automatic annotation of videos. We considered two types of annotations, visual semantic roles and scene locations. For the visual semantic roles we applied our existing semantic role labeling system to a new dataset and a new set of roles, showing that the system can straightforwardly be applied to these new settings. To annotate scenes in a video with their locations, we developed a novel method that combined information in the text with information in the video. This method consisted of a scene cut detector that combined a text classifier with a shot cut detector, and of a location detector based on a hidden Markov model. To propagate the discovered locations to visually and textually similar scenes we proposed a topic model where the topics where reweighted with visual similarity.

Throughout our work we proposed and evaluated different types of features for every information extraction task. We also consistently compared generative and discriminative models. This showed that discriminative models outperform generative models when large sets of training are available (e.g. word sense disambiguation, semantic role labeling), but that generative models can have surprisingly high performance on small training sets, outperforming discriminative models (e.g. scene cut classification, location detection).

To summarize, we list all novel models and methods that have been developed in this thesis:

- A semi-supervised model for semantic role labeling.
- A semi-supervised multiple-mixtures model for semantic role labeling.
- The relative discounted Kneser-Ney smoothing method.
- The latent words language model (LWLM).
- A method that uses the LWLM for weakly supervised word sense disambiguation.
- A method that uses the LWLM for weakly supervised semantic role labeling.
- A method to predict the entities present in an image based on an analysis of the descriptive text, comprising
  - A method to compute the salience of entities in a text.
  - A method to compute the visualness of entities from the WordNet dictionary.
- A method for the detecting and labeling arguments of visual verbs for the annotation of actions in videos.
- A method for the automatic annotation of locations of scenes in a video, comprising

- A multimodal scene cut detector for videos.
- A method for the automatic detection of location phrases in the text.
- A multimodal method to propagate location phrases to similar scenes.

These models were all tested on annotated data (models for classification or annotation) or unseen texts (language models) and compared to state-of-the-art models where available.

Lessons learned

We started our work with the observation that the limited performance of supervised models for many information extraction tasks is due to the large variation and ambiguity of natural language. We then set off with the aim of finding a method that would leverage the information present in unlabeled examples. However simple to conceive intuitively, this task was more difficult than initially anticipated. A method for semi-supervised learning with hidden variables was found not to be suited for this task. A second method based on an advanced language model, was more successful. We reported significant gains in performance when training the models on a limited training set. On larger training sets however only limited gains were achieved. At this moment it is unclear whether a further improvement requires improving the language model or the method used to incorporate results of the language model in an information extraction method. The outstanding performance of the language model on predicting unseen texts seem to point to the latter.

We are convinced that our research has pointed to some interesting directions for future research, although we are probably still only scratching the surface of the full potential of weakly supervised models. Research on this topic will likely continue for some years to come, since also after two decades no clear single best method has been discovered that results in significant gains on a number of information extraction tasks.

From our research on weakly supervised multimodal methods we learned that a good method for information extraction is indispensable when trying to combine the two media, and that it is possible to extract detailed annotations of images and video from descriptive texts using appropriate information extraction methods.

Future work

Our directions of future work can be divided among a number of topics. First we would like to investigate two extensions to the latent words language model:

(1) learn similarities between phrases instead of between words, since the meaning of multi-word expressions (e.g. "United Nations", "touch down") can often not be determined from the meaning of the individual words. This would involve a dynamic method to split a sequence of words into individual phrases. This could be performed with the inside-outside algorithm (Lari and Young, 1990), although it will also have to be adapted for the large number of hidden variables. This could be performed with techniques similar to the ones we used to adapt the forward-backward algorithm for hidden Markov models. (2) We would like to extend the LWLM to take into account all words in the context, and not only the words in a window of length n. This would improve the synonyms learned for a word in a certain context since the general topic or domain of a text influences the relevant synonyms. In this context we would like to investigate a combination of the LWLM with latent Dirichlet allocation [Blei et al., 2003], since this model has proven its adequacy for modeling document level topics.

As described above we do not belief that the full potential of weakly supervised models has been reached. More specifically we think that more research on methods to use the latent words language model for supervised information extraction still has a large potential. Ideally these method would be sufficiently general to also be applicable to other language models, since intuitively any knowledge of the structure of language should be useful for an automatic analysis of this language.

In our second area of research, multimodal image annotation using descriptive texts, many advances are still possible. We would like to extend the appearance model with a method that captures cues in the text (e.g. "George Bush (third from left)") to further strengthen the prediction of entities likely to be seen in the image. Ideally these cues would be learned automatically without relying on an annotated corpus to be independent of a certain corpus or domain. We would also like to combine the appearance model with the semantic role labeling system used for visual verbs. For example to identify objects as arguments to these verbs one can incorporate the visualness measure, since this already gives a strong indication of objects that are likely to appear in the video. We already outlined one successful application of the detected actions on news images, but would like to extend this in the future to a larger number of actions in real life video. Finally we think that the described method for automatic annotation of locations in scenes describes an interesting framework that can be adapted to other types of annotations, such as for example to the annotation of stories in news broadcasts. A different type of annotation would only require to retrain the location detector on a new annotations (e.g. to extract headlines of news stories) and would possibly require different types of features extracted from the video, depending of the entities of interest.

Bibliography

S. P. Abney. *Semisupervised Learning for Computational Linguistics.* Chapman & Hal, 2007.

A. Amir, J. Argillander, M. Campbell, A. Haubold, G. Iyengar, S. Ebadollahi, F. Kang, M. R. Naphade, A. Natsev, J. R. Smith, J. Tešió, and T. Volkmer. IBM Research TRECVID-2005 Video Retrieval System. In *Proceedings of TRECVID 2005*, Gaithersburg, MD, 2005.

C. Baker, C. Fillmore, and J. Lowe. The Berkeley FrameNet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, volume 98. Montreal, Canada, 1998.

J. Baker. The DRAGON system–An overview. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 23(1):24–29, 1975.

S. Baluja. Probabilistic modeling for face orientation discrimination: Learning from labeled and unlabeled data. *Advances in Neural Information Processing Systems*, pages 854–860, 1999.

L. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi-supervised learning on large graphs. In *Proceedings of the 17th Conference on Learning Theory.* Springer, 2004.

T. C. Bell, J. G. Cleary, and I. H. Witten. *Text Compression.* Prentice Hall, N.J., 1990.

R. Bellman and S. Dreyfus. *Applied dynamic programming.* Princeton University Press, 1962.

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

T. L. Berg, A. C. Berg, J. Edwards, and D. Forsyth. Who's in the Picture? In *Neural Information Processing Systems*, pages 137–144, 2004.

A. Berger, V. Pietra, and S. Pietra. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):71, 1996.

S. Bethard, H. Yu, A. Thornton, V. Hatzivassiloglou, and D. Jurafsky. Automatic extraction of opinion propositions and their holders. In *Proceedings of the 2004 AAAI Spring Symposium on Exploring Attitude and Affect in Text*, pages 22–24, 2004.

D. Bikel, R. Schwartz, and R. Weischedel. An algorithm that learns what's in a name. *Machine learning*, 34(1):211–231, 1999.

C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer New York, 2006.

E. Black. An experiment in computational discrimination of english word senses. *IBM Journal of Research and Development*, 32(2):185–194, 1988.

W. Bledsoe and I. Browning. Pattern recognition and reading by machine. *AFIPS Joint Computer Conferences*, pages 225–232, 1959.

D. M. Blei and M. I. Jordan. Modeling annotated data. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 127–134, 2003.

D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

M. Blighe and N. O'Connor. MyPlaces: detecting important settings in a visual diary. In *Proceedings of the 2008 International Conference on Content-based Image and Video Retrieval*, pages 195–204. ACM, 2008.

H. Boas. Bilingual FrameNet dictionaries for machine translation. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 1364–1371, 2002.

B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, pages 144–152. ACM New York, NY, USA, 1992.

G. Bouchard and B. Triggs. The tradeoff between generative and discriminative classifiers. In *IASC International Symposium on Computational Statistics (COMPSTAT)*, pages 721–728, 2004.

T. Brants. TnT–a statistical part-of-speech tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*, pages 224–231, 2000.

P. Brown, J. Cocke, S. Pietra, V. Pietra, F. Jelinek, J. Lafferty, R. Mercer, and P. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):85, 1990.

P. Brown, R. Mercer, V. Della Pietra, and J. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

R. Bruce. Semi-supervised learning using prior probabilities and EM. In *International Joint Conference of AI Workshop on Text Learning: Beyond Supervision, Seattle, Washington*, 2001.

C. Callison-Burch, D. Talbot, and M. Osborne. Statistical machine translation with word-and sentence-aligned parallel corpora. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 175. Association for Computational Linguistics, 2004.

V. Castelli and T. Cover. On the exponential value of labeled samples. *Pattern Recognition Letters*, 16(1):105–111, 1995.

V. Castelli and T. Cover. The relative value of labeled and unlabeled samples in pattern recognition with an unknown mixing parameter. *IEEE Transactions on Information Theory*, 42(6 Part 2):2102–2117, 1996.

S. Chang, J. Smith, M. Beigi, and A. Benitez. Visual information retrieval from large distributed online repositories. *Communications of the ACM*, 40(12):63–71, 1997.

O. Chapelle and A. Zien. Semi-supervised classification by low density separation. *Encyclopedia of Biostatistics*, 34:77–86, 2005.

O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

E. Charniak. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the National Conference on Artificial Intelligence*, pages 598–603. John Wiley & Sons, 1997.

L. Chen, Y. Lai, and H. Mark Liao. Movie scene segmentation using background information. *Pattern Recognition*, 41(3):1056–1065, 2008.

S. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318. Association for Computational Linguistics Morristown, NJ, USA, 1996.

H. Chernoff and E. Lehmann. The use of maximum likelihood estimates in $\chi 2$ tests for goodness of fit. *The Annals of Mathematical Statistics*, pages 579–586, 1954.

K. Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the second conference on Applied natural language processing*, volume 136, 1988.

M. Ciaramita and Y. Altun. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 594–602. Association for Computational Linguistics, 2006.

S. Clark, J. R. Curran, and M. Osborne. Bootstrapping POS taggers using unlabelled data. In *Proceedings of the Seventh CoNLL Conference*, pages 49–55. Association for Computational Linguistics Morristown, NJ, USA, 2003.

M. Collins and Y. Singer. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 189–196, 1999.

R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

A. Colmerauer. Total precedence relations. *Journal of the ACM*, 17(1):14–30, 1970.

T. Cour, C. Jordan, E. Miltsakaki, and B. Taskar. Movie/Script: Alignment and Parsing of Video and Text Transcription. In *ECCV '08: Proceedings of the 10th European Conference on Computer Vision*, pages 158–171, Berlin, Heidelberg, 2008. Springer-Verlag.

T. Cover and J. Thomas. *Elements of Information Theory.* Wiley-Interscience, 2006.

F. Cozman and I. Cohen. *Risks of Semi-Supervised Learning.* MIT Press, Cambridge, MA, 2006.

F. Cozman, I. Cohen, M. Cirelo, et al. Semi-supervised learning of mixture models. In *Proceedings of the Twentieth International Conference of Machine Learning*, pages 99–108, 2003.

W. B. Croft and J. Lafferty. *Language Modeling for Information Retrieval.* Kluwer Academic Publishers, Boston, MA, 2003.

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, 2002.

D. Cutting, J. Kupiec, J. Pedersen, and P. Sibun. A practical part-of-speech tagger. In *Proceedings of the third conference on Applied natural language processing*, pages 133–140. Association for Computational Linguistics Morristown, NJ, USA, 1992.

R. Datta, D. Joshi, J. Li, and J. Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Comput. Surv.*, 40(2):1–60, 2008. ISSN 0360-0300.

R. De Busser, R. Angheluta, and M.-F. Moens. Semantic case role detection for information extraction. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 1198–1202. Association for Computational Linguistics, 2002.

B. Decadt, V. Hoste, W. Daelemans, and A. Van den Bosch. Gambl, genetic algorithm optimization of memory-based wsd. In *Proceedings of the 3th International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (Senseval-3)*, pages 108–112, 2004.

M. DeGroot. *Optimal Statistical Decisions*. McGraw-Hill, New York, 1970.

A. Dempster, N. Laird, D. Rubin, et al. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

K. Deschacht and M.-F. Moens. Text analysis for automatic image annotation. In *Proceedings of the 45th Annual Meeting of the Assocation for Computational Linguistics*, pages 1000–1007, 2007.

K. Deschacht, M.-F. Moens, and W. Robeyns. Cross-media entity recognition in nearly parallel visual and textual documents. In *Proceedings of the 8th RIAO Conference on Large-Scale Semantic Access to Content (Text, Image, Video and Sound)*. CMU, 2007.

G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel. The automatic content extraction (ACE) program–tasks, data, and evaluation. In *Proceedings of 4th International Conference on Language Resources and Evaluation*, pages 837–840, 2004.

A. C. Doyle. The adventure of the copper beeches. *The strand magazine: an illustrated monthly*, 3:613–628, 1891.

O. Duchenne, I. Laptev, J. Sivic, F. Bach, and J. Ponce. Automatic Annotation of Human Actions in Video. In *Proceedings of the 12th IEEE Intenational Conference on Computer Vision*, 2009.

T. Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational linguistics*, 19(1):74, 1993.

J. B. Estoup. *Gammes Stenographiques*. Institut Stenographique de France, Paris, 1916.

M. Everingham, J. Sivic, and A. Zisserman. Hello! My name is... Buffy – automatic naming of characters in TV video. In *Proceedings of the British Machine Vision Conference*, volume 2, 2006.

C. Fellbaum. *WordNet: An Electronic Lexical Database*. The MIT Press, 1998.

V. Ferrari, M. Marín-Jiménez, and A. Zisserman. Progressive Search Space Reduction for Human Pose Estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

C. J. Fillmore. The case for case. In E. Bach and R. Harms, editors, *Universals in Linguistic Theory*. Rinehart & Winston, 1968.

J. Firth. *Papers in Linguistics, 1934-1951*. Oxford University Press, 1957.

S. Flank, P. Martin, A. Balogh, and J. Rothey. Photofile: A digital library for image retrieval. In *Multimedia Computing and Systems, 1995., Proceedings of the International Conference on*, pages 292–295, 1995.

W. N. Francis. A standard sample of present-day English for use with digital computers. Report to the U.S Office of Education on Cooperative Research Project No. E-007. Brown University, Providence RI, 1964.

H. Fürstenau and M. Lapata. Semi-supervised semantic role labeling. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 220–228, Athens, Greece, 2009. Association for Computational Linguistics.

D. Gildea and D. Jurafsky. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288, 2002.

S. Goldwater and T. Griffiths. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Assocation for Computational Linguistics*, volume 45, page 744, 2007.

J. T. Goodman. A bit of progress in language modeling, extended version. Technical report, Microsoft Research, 2001.

Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. *Advances in Neural Information Processing Systems*, 17:529–536, 2004.

G. Grefenstette. *Explorations in Automatic Thesaurus Discovery*. Springer, 1994.

T. Griffiths, M. Steyvers, D. Blei, and J. Tenenbaum. Integrating topics and syntax. *Advances in Neural Information Processing Systems*, 17:537–544, 2005.

R. Grishman and J. Sterling. Generalizing automatically generated selectional patterns. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 742–747, 1994.

R. Grishman and B. Sundheim. Message understanding conference-6: A brief history. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*, pages 466–471. Association for Computational Linguistics Morristown, NJ, USA, 1996.

B. Grosz et al. The representation and use of focus in a system for understanding dialogs. In *Proceedings of the fifth international joint conference on artificial intelligence*, pages 67–76, 1977.

J. Gruber. *Studies in Lexical Relations.* Indiana University Linguistics Club, 1970.

M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid. Automatic face naming with caption-based supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

M. Guillaumin, T. Mensink, J. Verbeek, C. Schmid, I. LEAR, and L. Kuntzmann. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *Proceedings of the 12th IEEE Intenational Conference on Computer Vision*, 2009.

M. Halliday. *An Introduction to Functional Grammar (second edition).* Edward Arnold, London, 1994.

Z. S. Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

Z. S. Harris. *String Analysis of Sentence Structure.* Mouton & Co, The Hague, 1962.

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

B. Hayes. The Web of Words. *American Scientist*, 87(2):108–112, March-April 1999.

S. He and D. Gildea. Self-training and co-training for semantic role labeling: Primary Report. Technical report, 2006. TR 891.

M. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics Morristown, NJ, USA, 1992.

M. Héritier, S. Foucher, and L. Gagnon. Key-Places Detection and Clustering in Movies Using Latent Aspects. In *ICIP*, 2007.

J. R. Hobbs. Pronoun resolution. *SIGART Bull.*, (61):28–28, 1977. ISSN 0163-5719.

G. Huang, M. Ramesh, T. Berg, and E. Learned Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, 2007.

N. Ide and J. Véronis. Introduction to the special issue on word sense disambiguation: the state of the art. *Computational Linguistics*, 24(1):2–40, 1998.

R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1):79–87, 1991.

V. Jain, E. Learned-Miller, and A. McCallum. People-LDA: Anchoring topics to people using face recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, 2007.

F. Jelinek, L. Bahl, and R. Mercer. Design of a linguistic statistical decoder for the recognition of continuous speech. *IEEE Transactions on Information Theory*, 21(3):250–256, 1975.

F. Jelinek and R. L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, Amsterdam, The Netherlands, 1980.

F. Jiao, S. Wang, C. Lee, R. Greiner, and D. Schuurmans. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 209–216. Association for Computational Linguistics Morristown, NJ, USA, 2006.

L. Jie, B. Caputo, and V. Ferrari. Who's doing what: Joint modeling of names and verbs for simultaneous face and pose annotation. In NIPS, editor, *Advances in Neural Information Processing Systems*, NIPS. NIPS, December 2009.

T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International Conference on Machine Learning*, 1999.

R. Johansson, A. Berglund, M. Danielsson, and P. Nugues. Automatic text-to-scene conversion in the traffic accident domain. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 19, page 1073. Lawrence Erlbaum Associates, 2005.

M. Johnson. Why doesn't EM find good HMM POS-taggers. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 296–305, 2007.

D. Jurafsky and J. H. Martin. *Speech and Language Processing*. Prentice Hall, 2008.

J. Kamps and M. Marx. Words with Attitude. In *Proceedings of the 1st International Conference on Global WordNet*, pages 332–341, India, 2002.

R. Kate and R. Mooney. Semi-supervised learning for semantic parsing using support vector machines. In *Proceedings of the Joint Conference of Human Language Technologies and the North American Chapter of the Association for Computational Linguistics*, pages 81–84, 2007.

J. J. Katz and J. A. Fodor. The structure of a semantic theory. *Language*, 39(2): 170–210, 1963.

M. Kay. Algorithm schemata and data structures in syntactic processing. In *Proceedings of the Nobel Symposium on Text Processing*, 1980.

M. Kemighan, K. Church, and W. Gale. A spelling correction program based on a noisy channel model. In *Proceedings of the Thirteenth International Conference on Computational Linguistics*, pages 205–210, 1990.

A. Kilgarriff. Senseval: An exercise in evaluating word sense disambiguation programs. In *Proceedings of the International Conference on Language Resources and Evaluation*, pages 581–588, 1998.

K. Kipper, H. Dang, and M. Palmer. Class-based construction of a verb lexicon. In *Proceedings of the National Conference on Artificial Intelligence*, pages 691–696. Menlo Park, CA, 2000.

T. Kliegr, K. Chandramouli, J. Nemrava, V. Svatek, and E. Izquierdo. Combining image captions and visual analysis for image concept classification. In *Proceedings of the 9th International Workshop on Multimedia Data Mining*, pages 8–17. ACM, 2008.

U. Kohomban and W. Lee. Learning semantic classes for word sense disambiguation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 34–41. Association for Computational Linguistics, 2005.

T. Koo, X. Carreras, and M. Collins. Simple semi-supervised dependency parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 595–603, 2008.

V. Krishnamurthy and J. Moore. On-line estimation of hidden Markov model parameters based on the Kullback-Leibler information measure. *IEEE Transactions on Signal Processing*, 41(8):2557–2573, 1993.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, 2001.

S. Landes, C. Leacock, and R. I. Tengi. Building Semantic Concordances. In C. Fellbaum, editor, *WordNet: An Electronic Lexical Database*. The MIT Press, 1998.

I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

K. Lari and S. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer speech and Language*, 4(1):35–56, 1990.

N. Lawrence and M. Jordan. Semi-supervised learning via Gaussian processes. *Advances in Neural Information Processing Systems*, 17:753–760, 2005.

L. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, 2006.

C. W. Leong and R. Mihalcea. Explorations in automatic image annotation using textual features. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 56–59, Suntec, Singapore, August 2009. Association for Computational Linguistics.

M. Lesk. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26. ACM New York, NY, USA, 1986.

B. Levin. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press, 1993.

L.-J. Li, R. Socher, and L. Fei-Fei. Towards Total Scene Understanding: Classification, Annotation and Segmentation in an Automatic Framework. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

W. Li and A. McCallum. Semi-supervised sequence modeling with syntactic topic models. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 813. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

J.-H. Lim, Y.-S. Hwang, S.-Y. Park, and H.-C. Rim. Semantic role labeling using maximum entropy model. In *Proceedings of the Eighth Conference on Computational Natural Language Learning*, pages 122–125, Boston, Massachusetts, USA, 2004. ACL.

D. Lin. Using syntactic dependency as local context to resolve word sense ambiguity. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, volume 35, pages 64–71. ACL, 1997.

D. Lin. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, pages 296–304, 1998a.

D. Lin. An Information-Theoretic Definition of Similarity. In *Proceedings of the 15th International Conference on Machine Learning*, 1998b.

G. Mann and A. McCallum. Simple, robust, scalable semi-supervised learning via expectation regularization. In *Proceedings of the 24th International Conference on Machine learning*, pages 593–600. ACM Press New York, USA, 2007.

C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 2002.

M. Marcus, B. Santorini, and M. Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1994.

M. Marszałek, I. Laptev, and C. Schmid. Actions in context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

M. Masterman. The thesaurus in syntax and semantics. *Mechanical Translation*, (4):1–2, 1957.

A. McCallum and W. Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Seventh Conference on Natural Language Learning (CoNLL)*, 2003.

J. McCarthy and W. Lehnert. Using decision trees for coreference resolution. *Arxiv preprint cmp-lg/9505043*, 1995.

D. McClosky, E. Charniak, and M. Johnson. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, 2006.

D. Mehay, R. De Busser, and M.-F. Moens. Labeling generic semantic roles. In *Proceedings of the Sixth International Workshop on Computational Semantics*, pages 175–188. Tilburg University, 2005.

G. Melli, Y. Wang, Y. Liu, M. Kashani, Z. Shi, B. Gu, A. Sarkar, and F. Popowich. Description of SQUASH, the SFU question answering summary handler for the DUC-2005 summarization task. In *Proceeding of Document Understanding Conference (DUC-2005)*, 2005.

B. Merialdo. Tagging English text with a probabilistic model. *Computational linguistics*, 20(2):155–171, 1994.

P. Merlo and L. van der Plas. Abstraction and generalisation in semantic role labels: PropBank, VerbNet or both? In *Proceedings of the 47th Annual Meeting of the ACL and the 4th International Conference on Natural Language Processing of the AFNLP*, pages 288–296, Suntec, Singapore, August 2009. Association for Computational Linguistics.

N. Metropolis and S. Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949. ISSN 01621459.

R. Mihalcea and E. Faruque. Senselearner: Minimally supervised word sense disambiguation for all words in open text. In *Proceedings of the 3th International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (Senseval-3)*, 2004.

A. Mikheev. Automatic Rule Induction for Unknown-Word Guessing. *Computational Linguistics*, 23(3):405–423, 1997.

S. Miller, J. Guinness, and A. Zamanian. Name tagging with word clusters and discriminative training. In *Proceedings of 2004 Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics Annual Meeting*, volume 4, 2004.

E. Miltsakaki, R. Prasad, A. Joshi, and B. Webber. The Penn discourse treebank. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, 2004.

C. Mitchell, M. Harper, L. Jamieson, et al. On the complexity of explicit duration HMM's. *IEEE transactions on speech and audio processing*, 3(3):213–217, 1995.

M.-F. Moens. Using Patterns of Thematic Progression for Building a Table of Content of a Text. *Journal of Natural Language Engineering*, 14(2):145–172, 2008.

F. Monay and D. Gattica-Perez. On image auto-annotation with latent space models. In *ACM Multimedia*, pages 275 – 278, 2003.

R. Morante, W. Daelemans, and V. Van Asch. A combined memory-based semantic role labeler of english. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 208–212. Association for Computational Linguistics, 2008.

Y. Mori, H. Takahashi, and R. Oka. Automatic Word Assignment to Images Based on Image Division and Vector Quantization. In *Proceedings of the RIAO Conference on Content-Based Multimedia Information Access*, Paris, April 12-14 2000.

R. Nag, K. Wong, and F. Fallside. Script recognition using hidden Markov models. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 11, 1986.

S. Narayanan and S. Harabagiu. Question answering based on semantic structures. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 693–703, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

H. Ney, U. Essen, and R. Kneser. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech and Language*, 8(1):1–38, 1994.

A. Ng and M. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in Neural Information Processing Systems*, 2:841–848, 2002.

K. Ni, A. Kannan, A. Criminisi, and J. Winn. Epitomic location recognition. In *Computer Vision and Pattern Recognition*, 2008.

K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. In *IJCAI-99 workshop on Machine Learning for Information Filtering*, pages 61–67, 1999.

K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled Documents using EM. *Machine Learning*, 39(2):103–134, 2000.

K. Nigam, A. McCallum, and T. Mitchell. Semi-supervised text classification using EM. In *Semi-Supervised Learning. MIT Press, Boston, MA*, London, England, 2006. MIT Press.

J. Nivre, J. Hall, and J. Nilsson. MaltParser: A datadriven parser-generator for dependency parsing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 2216–2219, 2006.

A. Oliva and A. Torralba. Building the gist of a scene: The role of global image features in recognition. *Progress in Brain Research*, 155:23–36, 2006.

D. Ozkan and P. Duygulu. A graph based approach for naming faces in news photos. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1477–1482, 2006.

M. Palmer, D. Gildea, and P. Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106, 2005.

M. Paramita, M. Sanderson, and P. Clough. Diversity in photo retrieval: overview of the ImageCLEFPhoto task 2009. *CLEF working notes*, 2009, 2009.

J. Pearl and G. Shafer. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan Kaufmann San Mateo, CA, 1988.

T. Pedersen, S. Patwardhan, and J. Michelizzi. WordNet::Similarity - Measuring the Relatedness of Concepts. In *The Proceedings of Fifth Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-04)*, Boston, May 2004.

F. Pereira, N. Tishby, and L. Lee. Distributional clustering of English words. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 183–190. Association for Computational Linguistics Morristown, NJ, USA, 1993.

F. Pereira and D. Warren. Parsing as deduction. In *Proceedings of the 21st annual meeting on Association for Computational Linguistics*, pages 137–144. Association for Computational Linguistics Morristown, NJ, USA, 1983.

P. T. Pham, M.-F. Moens, and T. Tuytelaars. Cross-media alignment of names and faces. *IEEE Transactions on Multimedia*, 12:13–27, 2010.

D. Pierce and C. Cardie. Limitations of co-training for natural language learning from large datasets. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 1–9. Association of Computational Linguistics, 2001.

S. Pradhan, W. Ward, K. Hacioglu, J. Martin, and D. Jurafsky. Shallow semantic parsing using support vector machines. In *Proceedings of the Joint Conference of Human Language Technologies and the North American Chapter of the Association for Computational Linguistics*, 2004.

S. Pradhan, E. Loper, D. Dligach, and M. Palmer. Semeval-2007 task-17: English lexical sample, srl and all words. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 87–92, 2007.

J. Pustejovsky, P. Hanks, R. Sauri, A. See, R. Gaizauskas, A. Setzer, D. Radev, B. Sundheim, D. Day, L. Ferro, et al. The timebank corpus. In *Corpus Linguistics*, pages 647–656, 2003.

A. Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142. Association for Computational Linguistics, 1996.

A. Ratnaparkhi. *Maximum entropy models for natural language ambiguity resolution.* PhD thesis, Philadelphia, PA, USA, 1998.

R. Redner and H. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26:195, 1984.

B. Roark and E. Charniak. Noun-phrase co-occurrence statistics for semi-automatic semantic lexicon construction. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International*

Conference on Computational Linguistics, Volume 2, pages 1110–1116, Montreal, Quebec, Canada, August 1998. Association for Computational Linguistics.

Y. Rubner, C. Tomasi, and L. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

P. Sankar K., C. V. Javahar, and A. Zisserman. Subtitle-free Movie to Script Alignment. In *British Machine Vision Conference*, 2009.

A. Sarkar. Applying co-training methods to statistical parsing. In *Proceedings of the second Annual Meeting of the NAACL*, pages 95–102, 2001.

S. Satoh, Y. Nakamura, and T. Kanade. Name-It: Naming and Detecting Faces in News Videos. *IEEE MultiMedia*, 6(1):22–35, January-March 1999.

F. Schaffalitzky and A. Zisserman. Automated Location Matching in Movies. *Computer Vision and Image Understanding*, 92(2-3):236–264, 2003.

R. C. Schank and R. P. Abelson. *Scripts, Plans, Goals and Understanding: An Inquiry into Human Knowledge Structures.* L. Erlbaum, Hillsdale, NJ, 1977.

F. Schroff, C. L. Zitnick, and S. Baker. Clustering videos by location. In *Proceedings of the British Machine Vision Conference*, 2009.

M. Seeger. Learning with labeled and unlabeled data. Technical report, Institute for Adaptive and Neural Computation, University of Edinburgh, 2002.

B. Shahshahani and D. Landgrebe. The effect of unlabeled samples in reducing the small sample size problem and mitigating the Hughes phenomenon. *IEEE Transactions on Geoscience and Remote Sensing*, 32(5):1087–1095, 1994.

D. Shen and M. Lapata. Using semantic roles to improve question answering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 12–21, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

L. Shue and S. Dey. Complexity reduction in fixed-lag smoothing for hidden Markovmodels. *IEEE Transactions on Signal Processing*, 50(5):1124–1132, 2002.

F. Smadja. Retrieving collocations from text: Xtract. *Computational linguistics*, 19(1):143–177, 1994.

A. Smeaton and I. Quigley. Experiments on using semantic distances between words in image caption retrieval. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 174–180. ACM New York, NY, USA, 1996.

A. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on pattern analysis and machine intelligence*, 22(12):1349–1380, 2000.

N. Smith and J. Eisner. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 354–362. Association for Computational Linguistics Morristown, NJ, USA, 2005.

R. Snow, D. Jurafsky, and A. Ng. Learning syntactic patterns for automatic hypernym discovery. *Advances in Neural Information Processing Systems*, 17: 1297–1304, 2005.

B. Snyder and M. Palmer. The English all-words task. In *Proceedings of the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (Senseval-3)*, volume 3, pages 41–43, 2004.

R. Srihari and C. Baltus. Combining statistical and syntactic methods in recognizing handwritten sentences. In *AAAI Symposium: Probabilistic Approaches to Natural Language*, pages 121–127, 1992.

S. Stevenson and X. Carreras, editors. *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*. Association for Computational Linguistics, Boulder, Colorado, June 2009.

M. Surdeanu, S. Harabagiu, J. Williams, and P. Aarseth. Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 8–15, 2003.

M. Surdeanu, R. Johansson, A. Meyers, L. Marquez, and J. Nivre. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*, 2008.

R. Swier and S. Stevenson. Unsupervised semantic role labelling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 95–102, 2004.

M. Tang, X. Luo, and S. Roukos. Active learning for statistical natural language parsing. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 120–127. Association for Computational Linguistics Morristown, NJ, USA, 2001.

C. Thompson, R. Levy, and C. Manning. A generative model for framenet semantic role labeling. In *Proceedings of the 14th European Conference on Machine Learning*, Cavtat-Dubrovnik, Croatia, 2006.

S. Tratz, A. Sanfilippo, M. Gregory, A. Chappell, C. Posse, and P. Whitney. PNNL: A supervised maximum entropy approach to word sense disambiguation. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 264–267. Association for Computational Linguistics, 2007.

P. Turney. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 417–424, 2002.

Twiz TV. Buffy, The Vampire Slayer Transcripts. *http://www.twiztv.com/scripts/buffy/*.

A. Vailaya, M. Figueiredo, A. Jain, and H. Zhang. Image classification for content-based indexing. *IEEE Transactions on Image Processing*, 10(1):117–130, 2001.

V. Vapnik. The nature of statistical learning, 1995.

A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

J. Vogel and B. Schiele. Semantic modeling of natural scenes for content-based image retrieval. *International journal of computer vision*, 72(2):133–157, 2007.

Q. Wang and D. Schuurmans. Improved estimation for unsupervised part-of-speech tagging. In *Proceedings of the IEEE International Conference on Natural Language Processing and Knowledge Engineering*, 2005.

Q. I. Wang, D. Schuurmans, and D. Lin. Semi-supervised convex training for dependency parsing. In *Proceedings of ACL-08: HLT*, pages 532–540, Columbus, Ohio, June 2008. Association for Computational Linguistics.

W. Weaver. Translation. In W. N. Locke and A. D. Booth, editors, *Machine Translation of Languages*, pages 15–23. John Wiley & Sons, New York, 1955.

T. Westerveld. Image Retrieval: Content versus Context. In *Proceedings of the RIAO Conference on Content-Based Multimedia Information Access*, pages 276–284, April 2000. ISBN 2-905450-07-X.

T. Westerveld, J. C. van Gemert, R. Cornacchia, D. Hiemstra, and A. de Vries. An Integrated Approach to Text and Image Retrieval. In *Proceedings of TRECVID 2005*, Gaithersburg, MD, 2005.

T. Winograd. *Understanding Natural Language.* Academic Press, 1972.

G. Wyszecki and W. S. Styles. *Color Science: Concepts and Methods, Quantitative Data and Formulae.* Wiley, New York, 1982.

D. Xia, F. Wu, W. hao Liu, and H. wang Zhang. Image interpretation: mining the visible and syntactic correlation of annotated words. *Journal of Zhejiang University Science A*, 10(12):1759–1768, 2009.

J. Yang, M. Chen, and A. Hauptmann. Finding person x: Correlating names with visual appearances. *Lecture notes in computer science*, pages 270–278, 2004.

J. Yang, R. Yan, and A. Hauptmann. Multiple instance learning for labeling faces in broadcasting news video. In *Proceedings of the 13th annual ACM international conference on Multimedia*, page 40. ACM, 2005.

D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics Morristown, NJ, USA, 1995.

S. Yu and H. Kobayashi. An efficient forward-backward algorithm for an explicit-duration hidden Markov model. *IEEE Signal Processing Letters*, 10(1):11–14, 2003.

J. Yuan, H. Wang, L. Xiao, W. Zheng, J. Li, F. Lin, and B. Zhang. A formal study of shot boundary detection. *IEEE transactions on circuits and systems for video technology*, 17(2):168–186, 2007.

L. Zelnik-Manor and P. Perona. Self-Tuning Spectral Clustering. *Advances in Neural Information Processing Systems*, 17(1601-1608):16–23, 2004.

Y. Zhai and M. Shah. A General Framework for Temporal Video Scene Segmentation. In *Tenth IEEE International Conference on Computer Vision*, volume 2, 2005.

H. Zhao, W. Chen, J. Kazama, K. Uchimoto, and K. Torisawa. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 61–66. Association for Computational Linguistics, 2009.

S. Zhu and Y. Liu. Video scene segmentation and semantic representation using a novel scheme. *Multimedia Tools Appl.*, 42(2):183–205, 2009. ISSN 1380-7501.

X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

G. K. Zipf. *Human Behaviour and the Principle of Least Effort.* Addison-Wesley, Cambridge, Massachusetts, 1949.

Publication list

Conferences and workshops, first author

- Koen Deschacht and Marie-Francine Moens. *Efficient Hierarchical Entity Classification Using Conditional Random Fields.* In proceedings of the 2nd Workshop on Ontology Learning and Population, Sydney, 2006.
- Koen Deschacht and Marie-Francine Moens. *Efficient Hierarchical Entity Classification Using Conditional Random Fields*, 18th Belgian-Dutch Conference on Artificial Intelligence, Namur, 2006.
- Koen Deschacht and Marie-Francine Moens. *Text Analysis for Automatic Image Annotation.* In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, Prague, 2007.
- Koen Deschacht and Marie-Francine Moens. Text Analysis for Automatic Image Annotation, In Proceedings of the 19th Belgian-Dutch Conference on Artificial Intelligence (Dastani, M. and de Jong, E., eds.), pp. 260-267, The Netherlands, 2007.
- Koen Deschacht, Marie-Francine Moens and Wouter Robeyns. *Cross-Media Entity Recognition in Nearly Parallel Visual and Textual Documents.* In Proceedings of the 8th RIAO conference on Large-Scale Semantic Access to Content (Text, Image, Video and Sound), USA, 2007.
- Koen Deschacht and Marie-Francine Moens. *Finding the Best Picture: Cross-Media Retrieval of Content.* In C. Macdonald, I. Ounis, V. Plachouras & I. Ruthven (Eds.) Proceedings of the 30th European Conference on Information Retrieval. Lecture Notes in Computer Science 4956 (pp.539-546), Springer, 2008.
- Koen Deschacht and Marie-Francine Moens. *The Latent Words Language Model.* In Proceedings of the 18th Annual Belgian-Dutch Conference on Machine Learning (Benelearn 09), Tilburg, 2009.

- Koen Deschacht and Marie-Francine Moens. *Using the Latent Words Language Model for Semi-Supervised Semantic Role Labeling.* In proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP 2009), Singapore, August 7, 2009.

Conferences and workshops

- Erik Boiy, Pieter Hens, Koen Deschacht and Marie-Francine Moens. *Automatic Sentiment Analysis of On-line Text.* In Proceedings of the 11th International Conference on Electronic Publishing, Openness in Digital Publishing: Awareness, Discovery & Access, June 13-15, Vienna Austria, 2007.
- Piek Vossen, Katja Hoffman, Maarten De Rijke, Erik Tjong, Sang Kim and Koen Deschacht. *The Cornetto Database: Architecture and User-Scenarios.* In proceedings of the 7th Dutch-Belgian Information Retrieval Workshop (DIR 2007), Leuven, Belgium, March 28-29, 2007
- Erik Boiy, Koen Deschacht and Marie-Francine Moens. *Learning Visual Entities and their Visual Attributes from Text Corpora.* In Proceedings of the 5th International Workshop on Text-based Information Retrieval, IEEE Computer Society Press, 2008.
- Phi The Pham, Koen Deschacht and Marie-Francine Moens. *Document Clustering with User Feedback.* In Eduard Hoenkamp, Martine De Cock and Véronique Hoste (Eds.), Proceedings of the 8th Dutch-Belgian Information Retrieval Workshop (pp. 73-80).Maastricht University, 2008.
- Javier Arias, Koen Deschacht and Marie-Francine Moens. *Content Extraction from Multilingual Web Pages.* In Aly R., Hauff, C. Hiemstra, D., Huybers, T. & De Jong, F. (Eds.), Proceedings of the 9th Dutch-Belgium Information Retrieval Workshop, University of Twente, 2009.

International journals

- Gert-Jan Poulisse, Marie-Francine Moens, Tomas Dekens and Koen Deschacht. *News Story Segmentation in Multiple Modalities.* Multimedia Tools and Applications (in press), 2010.

Patent applications

- Koen Deschacht & Marie-Francine Moens. *Method for the automatic determination of context dependent hidden word distributions.* Submitted to U.S. Patent and Trademark Office on November 18, 2009.

Articles submitted to international journals

- Koen Deschacht and Marie-Francine Moens. *The latent words language model*. Submitted to Computational Linguistics.
- Koen Deschacht and Marie-Francine Moens. *Weakly supervised learning for semantic role labeling*. Submitted to the Journal of Artificial Intelligence Research.

Appendix A

Distributed computing architecture

We have developed a distributed computing infrastructure in Java to perform a task in parallel on many computers. This infrastructure is robust, very easy to set-up and downloads automatically executable files and datasets and results results and exceptions to the commanding client. Furthermore it has been implemented for performance, with data caching, load balancing and automatically selection of the fastest computing clients. It has also been designed with the explicit goal of being easy to use in Java programs.

A.1 Usage

We have opted for the following design: a *commanding client* sends a number of jobs to the *job server*. The job server sends these jobs to one or multiple *computing clients* that execute every job in parallel. After every job has been executed, the results are send back to the job server who sends them in turn sends to the commanding client. We will describe the usage of our framework a bit more in detail

Remote job

The basic unit of work is the remote job. To create a new type of jobs, the user creates a subclass of the class `RemoteJob` and implements the abstract method `void execute(ComputingClient computingClient)`. In design terms this method would be called a hot-spot of our framework. Implementing this method, and making sure that the job and the data used by the job are serializable (to be send over the network), is the *only work* that need to be performed by a user of this framework. All other aspects are handled automatically.

Commanding client

The class that is used by the user to send and receive jobs, is the class `CommandingClient`. First a new object of this class is created using the constructor with the ip-address, or the dns names of the job server. Then a number of jobs are sent to the server with the method `int sendJobToServer(RemoteJob)`. The user then waits for the results of these jobs with the methods `JobWrapper waitForJob(int jobId)` or `JobWrapper waitForAnyJob()`. These methods will block until either the job with the given id is received, or any job is received. Both methods return a `JobWrapper` object that contains the job, any data generated by the job, and exceptions thrown by the job, if any. The class `CommandingClient` is the only class of the framework that is used by an end-user.

A.2 Execution of a job

Once the user has passed a job to the method `int sendJobToServer(RemoteJob)` of the class `CommandingClient`, the job is handled by the framework in a way that is completely hidden from the end-user. In this section we will see this process in a bit more detail. First the class `int sendJobToServer(RemoteJob)` serializes the job, i.e. it converts the job and any data it references to in a sequence of bytes. For this is uses the standard `ObjectOutputStream` class in the Java API. This sequence of bytes is then compressed with the standard `ZipOutputStream` class. Once the job (and any data it references) is compressed, it is send to the job server. The job server will try to find a computing client that is not executing a job. If multiple computing clients are available, it will send the job to the client with lowest average execution time per job. Although not exact, since the time taken by different jobs differs, in reality usually many similar jobs are send to the server, and this heuristic is usually able to select the fastest computing client available. If all computing clients are occupied it stores the job until a computing client becomes available. Once the job arrives at a computing client, it is unpacked (using `ZipInputStream` and `ObjectInputStream`), and its `void execute(ComputingClient computingClient)` method is executed. The job will now perform the tasks implemented by the user, until this method finishes. The job is then again serialized and compressed, together with data that was generated during the execution (if any). The computing client then passes the job to the job server who passes it to the commanding client. The end-user will then receive the job through either the `JobWrapper waitForJob(int jobId)` or the `JobWrapper waitForAnyJob()` method.

A.3 Automatic class loading

When a remote job is executed on the computing client, it will be most likely make use of classes other than the remote job (e.g. datastructures to hold results, methods designed for complex mathematical operations, ...). If these classes are not within the standard Java API, the virtual machine on the computing client will not know them. A possible solution would be to force the end-user to collect all the necessary class files in a library (e.g. a jar file) that is stored on every computing client. This solution would require the end-user to make a list of all class-files employed (which is a non-trivial task for all but the most simple programs), to collect them and to store them in a library.

We have chosen for a more user-friendly and flexible solution. In Java, when an object of a certain class is created, the virtual machine loads the class definition through a class loader. The standard class loader in Java is designed to look for classes in the class path on the file system. In our architecture we use a new class loader on the computing client. This loader is implemented as a subclass of the standard class loader, and it implements an extra method which checks whether the needed class files are present in the class path. If they are not found, it sends a message to the job server asking for this class. The job server passes this message to the correct commanding client, which locates the class definition on its class path, serializes it, and sends it to the job server who passes it on to the computing client. The computing client deserializes the class definition and adds it to the classes know by the virtual machine.

A.4 Robustness

We wanted to make sure that the distributed system is robust against computer failures. We have implemented a simple scheme that protects the systems agains failures of computing clients. When the job server sends a job to a computing client, it remembers the job that was send to this client. If for some reason the connection with the computing client is lost (e.g. due to a network failure), the job is passed to a different computing client. Another risk is when a computing client is overloaded and takes a very long time to finish a job, although otherwise reacting normally. As a precaution to this case we send a job to different computing clients (max 3), if other computing clients are idle and no other jobs need to be executed. From the moment one of the computing clients returns the executed job, the jobs on the other clients are terminated.

A.5 Security

Because the described architecture allows for the executing of arbitrary Java code on the computing clients, security is a serious concern. We have implemented a simple but effective security protocol based on a SSL connection with a private and public key pair. The job server holds the private part of the key. Both the computing and commanding clients have a copy of public part of the key. When a client connects to the server the public key is checked against the private part and if matched, the connection is established. If the public key does not match, or no public key is offered, the connection is immediately terminated. Once a connection is made, all data send on this connection is SSL-encrypted, thus offering a reliable protection against sniffing or a man-in-the-middle attack. As a further security measure the IP addresses of the commanding clients are logged, together with the number of executed jobs and the computing clients on which these jobs were executed.

An important weakness in this design is the distribution of the public key: every computer executing a computing or commanding client needs a copy of this key. If one of this clients however is compromised, and a malicious party obtains a copy of the public key, he gains full access to the system and thus to all computing clients.

A.6 Future work

We have build this distributed system mainly with the goal of being able to run the latent words language model in parallel on a large number of computers. Although we have also used the framework for several other distributed tasks, it has never been tested by users outside our research group. Questions that are not fully answered at this moment are: how many commanding clients can be the job server handle before becoming a bottleneck? How well can it cope with jobs that need large amounts of data? How secure is it against a targeted attack? How well does the system perform on different computer architectures (e.g. cluster computers, multiple processors, shared memory architectures, ...)?. We think that the most important weakness in the design at this moment is the hanlding of jobs of different commanding clients. If different commanding clients execute large amounts of jobs, the jobs are send to the computing clients on a first-come, first-serve basis. This however does not take into account the expected execution time of a job, or the amount of jobs that are already being executed for a particular commanding client. We plan to address this problem in future work.

Appendix B

Iterative line search

For the latent words language model we have a collection of v smoothing parameters $\gamma = [\gamma_1, ..., \gamma_v]$ with lower bounds $l = [l_1, ..., l_v]$ and upper bounds $u = [u_1, ..., u_v]$. We would like to select values for these parameters such that the model assigns maximal probability to a collection of unseen texts. In this section we develop a simple algorithm that finds the values for some parameters $\gamma = [\gamma_1, ..., \gamma_v]$ so that they represent a (possibly local) maximum for some objective function $f(.)$. Note that this algorithm is general in the sence that it can be used to optimize any objective function $f(.)$ and not only the probability of unseen text according to a language model.

The algorithm is shown in listing 4. This algorithm iterates over all the parameters as long as an improvement was made in the previous iteration (stored in variable G). In every iteration we test for every parameter γ_i whether an improvement can be made by increasing or decreasing this parameter with ϵ, taking into account the lower and upper bounds. If an improvement can be made, the parameter is set to this new value. We then try the next parameter and so on. Since we improve the objective function $f(.)$ in every step we are guaranteed to find a (local) maximum.

Although this algorithm is very simple, we found that the number of steps needed to find the optimal smoothing parameters for the latent words language model was sufficiently low (usually around 5 iterations over all the variables), given a proper value for ϵ was chosen. Of course more complex optimization methods (e.g. quasi-Newton methods), would most likely converge even faster.

Algorithm 4 .

```
Require: $\gamma = [\gamma_1, ..., \gamma_v]$, $l = [l_1, ..., l_v]$, $u = [u_1, ..., u_v]$ and $f(.)$
 1: $G \Leftarrow$ true
 2: $max \Leftarrow f(\gamma)$
 3: while $G$ do
 4:     $G \Leftarrow$ false
 5:     for $i = 1$ to $v$ do
 6:         $\gamma_i \Leftarrow \gamma_i - \epsilon$
 7:         if $\gamma_i > l_i$ then
 8:             $val_l = f(\gamma)$
 9:         else
10:             $val_l = -\infty$
11:         end if
12:         $\gamma_i \Leftarrow \gamma_i + 2\epsilon$
13:         if $\gamma_i < u_i$ then
14:             $val_r = f(\gamma)$
15:         else
16:             $val_r = -\infty$
17:         end if
18:         if $val_l > val_r$ AND $val_l > max$ then
19:             $\gamma_i \Leftarrow \gamma_i - 2\epsilon$
20:             $max \Leftarrow val_l$
21:             $G \Leftarrow$ true
22:         else if $val_r > val_l$ AND $val_r > max$ then
23:             $max \Leftarrow val_r$
24:             $G \Leftarrow$ true
25:         else
26:             $\gamma_i \Leftarrow \gamma_i - \epsilon$
27:         end if
28:     end for
29: end while
```

Appendix C

Computation of expected value of sequences of hidden words

In section 6.2.2 we have explained how we compute the expected value of a hidden word at a particular position as

$$P(h_j|\mathbf{w}_{train}, C^\tau) \simeq \frac{\sum_{\mathbf{h}^{j+\delta}_{j+\delta-n+2}} \gamma'(\mathbf{h}^{j+\delta}_{j+\delta-n+2}, h_j)}{P(\mathbf{w}_{train}|C^\tau)}$$

where the values $\gamma'(\mathbf{h}^{j+\delta}_{j+\delta-n+2}, h_j)$ were defined as the joint probability of observing the words $\mathbf{w}^i_1$, the sequence $\mathbf{h}^i_{i-n+2}$ and the hidden variable h_j. This value was computed by first passing the forward messages $\alpha(\mathbf{h}^i_{i-n+2})$ up to position j, and then passing the message $\gamma'(\mathbf{h}^{j+\delta}_{j+\delta-n+2}, h_j)$ for δ more positions. In this section we will show how we compute $P(\mathbf{h}^j_{j-n+2}|\mathbf{w}_{train}, C^\tau, \gamma^\tau)$, the probability of observing sequence $\mathbf{h}^j_{j-n+2}$ given observed words $\mathbf{w}_{test}$. We again define a new forward probability $\gamma(\mathbf{h}^i_{i-n+2}, \mathbf{h}^j_{j-n+2})$ which is the joint probability of observing the words $\mathbf{w}^i_1$, the sequence $\mathbf{h}^i_{i-n+2}$ and the sequence $\mathbf{h}^j_{j-n+2}$. This value is defined for $i \geq j$ and is given by

$$\gamma(\mathbf{h}^i_{i-n+2}, \mathbf{h}^j_{j-n+2}) = \begin{cases} P(w_i|h_i) \sum_{h_{i-n+1}} \alpha(\mathbf{h}^{i-1}_{i-n+1}) P(h_i|\mathbf{h}^{i-1}_{i-n+1}) & if\, i = j \\ P(w_i|h_i) \sum_{h_{i-n+1}} \gamma(\mathbf{h}^{i-1}_{i-n+1}, \mathbf{h}^j_{j-n+2}) P(h_i|\mathbf{h}^{i-1}_{i-n+1}) & if\, i > j \end{cases}$$

We can interpret $\gamma(\mathbf{h}^{i}_{i-n+2}, \mathbf{h}^{j}_{j-n+2})$ as a series of messages, where for every different value of $\mathbf{h}^{j}_{j-n+2}$, a series of messages $\gamma(\mathbf{h}^{i}_{i-n+2}, \mathbf{h}^{j}_{j-n+2})$ is passed from position j to the end of the sequence. A trimmed version of this variable is defined as

$$\gamma'(\mathbf{h}^{i}_{i-n+2}, \mathbf{h}^{j}_{j-n+2}) = \begin{cases} \sigma_{i-n+1} P(w_i|h_i) \sum_{h_{i-n+1}} trim(\alpha'(\mathbf{h}^{i-1}_{i-n+1})) P(h_i|\mathbf{h}^{i-1}_{i-n+1}) & if \\ \sigma_{i-n+1} P(w_i|h_i) \sum_{h_{i-n+1}} trim(\gamma'(\mathbf{h}^{i-1}_{i-n+1}, \mathbf{h}^{j}_{j-n+2})) P(h_i|\mathbf{h}^{i-1}_{i-n+1}) & if \end{cases}$$

The sum in this equation has time complexity $O(b|V|)$ and sorting the values has complexity $O(b|V|\log(b|V|))$. Doing so for every position in the sequence results in a time complexity of $O(N_u \times [b|V|\log(b|V|)])$. The final probability is then computed as

$$P(\mathbf{h}^{j}_{j-n+2}|\mathbf{w}_{train}, C^{\tau}) \simeq \frac{\sum_{\mathbf{h}^{N_u}_{N_u-n+2}} \gamma'(\mathbf{h}^{N_u}_{N_u-n+2}, \mathbf{h}^{j}_{j-n+2})}{P(\mathbf{w}_{train}|C^{\tau})}$$

This value can again be limited to the δ positions after j to reduce the time complexity

$$P(\mathbf{h}^{j}_{j-n+2}|\mathbf{w}^{N_t}_{1}, C^{\tau}) \simeq P(\mathbf{h}^{j}_{j-n+2}|\mathbf{w}^{j+\delta}_{1}, C^{\tau})$$

which then results in

$$P(\mathbf{h}^{j}_{j-n+2}|\mathbf{w}_{train}, C^{\tau}) \simeq \frac{\sum_{\mathbf{h}^{j+\delta}_{j+\delta-n+2}} \gamma'(\mathbf{h}^{j+\delta}_{j+\delta-n+2}, \mathbf{h}^{j}_{j-n+2})}{P(\mathbf{w}_{train}|C^{\tau})}$$

Since we need to compute this probability for every position in the sequence, the total time complexity is $O(N_u \times (1+\delta) \times (b|V| + b|V|\log(b|V|))$. We see how this time complexity is equivalent with the time complexity to compute the expected value for the individual hidden words. In fact, the entire algorithm is almost equivalent.