Unsupervised Natural Language Processing for Knowledge Extraction from Domain-specific Textual Resources

Von der Fakultät für Mathematik und Informatik
der Universität Leipzig
angenommene

Dissertation

zur Erlangung des akademischen Grades

Doctor rerum naturalium
(Dr. rer. nat.)

im Fachgebiet
Informatik

vorgelegt

von Dipl. Inf. Christian Hänig
geboren am 15. September 1982 in Leipzig

Berlin, den 25. Oktober 2012

Die Annahme der Dissertation wurde empfohlen von:
1. Prof. Dr. Gerhard Heyer, Universität Leipzig
2. Prof. Dr. Jonas Kuhn, Universität Stuttgart

Die Verleihung des akademischen Grades erfolgt mit Bestehen
der Verteidigung am 17.04.2013 mit dem Gesamtprädikat *magna cum laude*.

ACKNOWLEDGMENTS

I would like to thank everyone who supported this thesis in any way.

Special thanks go to Prof. Gerhard Heyer, my supervisor. He supported me and my work from the beginning with numerous discussions about the way to go. Occasionally, he gave me a nudge into the right direction providing a different perspective to problems. I particularly esteemed that he always had trust in my ideas and plans which gave me the essential confidence to finish this thesis.

Without my colleagues and friends at Daimler, this thesis would not have been possible. I want to thank Jürgen Franke for his tireless efforts to ensure proper working conditions for his students. He always supported me with everything I needed – valuable discussions about problems, extra hardware or just companionable jogging to get a clear head. His commitment went far beyond the professional level and I am deeply thankful for his friendly support.

I would also like to thank Martin Schierle, Dominik Wetzel, Robert Remus and Mathias Bank for our endless discussions, legendary coffee breaks and of course – our friendship. You shaped my time in Ulm and made it unforgettable.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AIM Corpus Automotive Internet Mining Corpus

AIM Eval Automotive Internet Mining Corpus Evaluation Corpus

CCL Common Cover Links

CCM Constituent-Context Model

CCS Constituent Chunk Score

CFG Context-free Grammar

CSTK Contiguous Subtree Kernel

CVS Contextual Valence Shifters

DARPA Defense Advanced Research Projects Agency

DG Dependency Grammar

DMV Dependency Model with Valence

DOP Data-Oriented Parsing

ER Entity Recognition

HMM Hidden Markov Model

IE Information Extraction

MDL Minimum Description Length

MLE Maximum Likelihood Estimation

NEGRA NEGRA Corpus

NER Named Entity Recognition

NLP Natural Language Processing

OWL Web Ontology Language

PCM Polarity Composition Model

PSG Phrase structure grammar

PTK Partial Tree Kernel

QUIS Corpus Quality Information System Corpus

QUIS Eval Quality Information System Evaluation Corpus

RDF Resource Description Framework

RE Relation Extraction

SSTK Sparse Subtree Kernel

STK Subset Tree Kernel

STTS Stuttgart-Tübingen Tagset

SVD Singular Value Decomposition

SVM Support Vector Machine

TiGer TiGer Corpus

TSA Tag Sequence Alignment

UBM Unlabeled Brackets Measure

URI Unsupervised Relation Identification

U-DOP Unsupervised Data-Oriented Parsing

WSJ Wall Street Journal Corpus

1

INTRODUCTION

Living in an information age guarantees nearly infinite amounts of available knowledge. A huge portion of this information is encoded in written form of natural language while many facts already have been stored in a structured form like in databases. Structured and easy accessible knowledge is integrated in many business processes and is a well-established source for analyses with various objectives (e. g. quality assurance or business innovations). Still, the tremendous knowledge hidden in textual data is kept out of most of such processes as the analysis of language is difficult and yet not considered a solved task. *Text Mining* methods are developed to process natural language data and to detect and extract interesting facts.

With the growing amount of knowledge that can be automatically extracted from textual data, companies gradually perceive the potential in analyzing the data to improve business processes. The Daimler AG initiated a Text Mining project to extract knowledge and useful facts out of numerous textual resources available within the automotive domain to improve corporate quality analysis regarding two different scopes.

The major task was to improve processes for classical quality assurance. Quality assurance processes of most companies are still dominated by structured data (e. g. part codes, labor operation codes, etc.) stored in databases. This data is mined employing data mining methods which proved to extract reliable facts and to reveal statistical abnormalities. As information is often stored in an encoded form, the data loses many details due to the abstraction and clas-

sification into a finite set of discrete codes. The aim of the Text Mining project was to transform unstructured data provided by repair orders into structured facts retaining the information density provided in the texts. Established data mining methods can be applied afterwards to induce further information along with newly developed sophisticated analysis methods properly considering the textual data's nature.

The second use case was *Automotive Internet Mining*. Many specialized internet fora and blogs exist throughout the World Wide Web containing facts and opinions about automobiles. This data's value originates in containing unasked and uncensored thoughts of the customers. It contains problem descriptions as in repair orders, opinions about certain features or even suggestions, in which way a car could be improved. As this information is available for all manufacturers, mining this data unveils information that is not obtainable by any other quality sensor.

The extraction of facts is divided into two major parts – entity recognition and the extraction of relations between entities. While most scientific research in these fields deal with newspaper text and concentrate on extraction of entities like persons, organizations, places and temporal expressions, this thesis deals with entities of the automotive domain. Components, locations all over vehicles, failure describing symptoms and corrective actions applied by technicians are some of the entities in question.

In order to analyze the requirements of different components for natural language processing, the major differences between both data sources – repair order texts and automotive internet forum posts – in terms of language will be explored. Dedicated solutions for each of various natural language processing tasks are developed and applied to both use cases. The major aim of this thesis is to develop a relation extraction algorithm concentrating on the scope automotive domain.

1.1 MOTIVATION

With the growing amount of data Information Extraction (IE) gains growing attention, too. Remarkable progress has been made in both of its major subtasks – Entity Recognition (ER) and Relation Extraction (RE). While ER deals with the detection of entities, aims RE on extraction of relation between different entities. In past and recent research, ER concentrates on extraction of Named Entities (namely Named Entity Recognition (NER)). As the scientific community focused on analyzing newspaper corpora (which is still true in most cases) the most studied named entities are person names, organizations, places and temporal expressions. For these entities several annotated corpora exist and are used for training and evaluation. The performance of state of the art systems on newspaper data achieves scores in the high nineties and is close to perfection. The scientific field of RE looks very similar. Highly sophisticated and linguistically grounded models exist yielding very strong results for relations between the aformentioned Named Entities (e. g. *is-a*, *is-CEO-of* or *X-acquired-Y*).

Although very good results are achieved in both subtasks, most of the applied approaches per se cannot be easily adapted to specialized domains. Research on domain adaption showed that data in some specialized fields show fundamental differences in language on all levels – starting with different morphology, terminology and syntax. The scope also differs regarding some semantic issues like entity and relation types. The automotive domain needs to focus on components and symptoms describing real world problems of the customers rather than extracting knowledge about names and places all over the world.

In order to develop a relation extraction system fulfilling all requirements of the automotive domain, characteristics of its domain language need to be considered and understood thoroughly. Analyses of state of the art systems will reveal the necessity of unsupervised algorithms that can adapt to textual data sources showing arbitrary linguistic characteristics. This thesis will present these analyses and unsupervised natural language processing methods and applies them to real world data and problems.

1.2 SCIENTIFIC CONTRIBUTION

A lot of effort has been put into development of unsupervised algorithms for Natural Language Processing (NLP). Nevertheless, many algorithms use supervised approaches to pre-annotate training data, but do not match certain criteria when adapting them to real world systems (e. g. memory consumption, run time or *black box* behavior). This thesis will deal with the requirements analysis of a dedicated automotive relation extraction system. Essential annotation steps will be identified and incorporated into a relation extraction system yielding state of the art results for two fundamentally different textual resources of the automotive domain.

The main contribution is the development of unsupervised algorithms for natural language processing. It will be shown, how unsupervised part-of-speech tagging can be improved by integration of disambiguation of high frequency words. Although disambiguation has been knowingly ignored due to only small potential for improvement on newspaper corpora, automotive corpora show a high portion of high frequency words yielding different syntactical functions.

An approach to unsupervised parsing will be presented. In contrast to most algorithms for grammar induction, additionally knowledge beyond induced grammars and parse trees is extracted. The presented algorithm recognizes different phrases bearing the same syntactical function and labels them appropriately. It also provides information about heads and modifiers that facilitates further processing like relation extraction. A verb detection module detects verbs of the language in question which supports sentence segmentation, nominal phrase detection and enables the possibility of unsupervised verb–argument detection.

For information extraction out of automotive corpora a syntactic relation extraction methodology is developed – relying on unsupervised pre-processing. The only supervised annotation step throughout the proposed relation extraction system is the NER module. Named Entity Recognition is achieved by application of a domain dependent thesaurus to ensure high quality results focusing on domain-specific terminology. The resulting relation extraction algorithm is

based on probabilistic relation extraction incorporating both syntactic and semantic features.

Part I

THEORETICAL ISSUES

2

TEXT MINING

During the last decade, many researchers tried to find a definition of what exactly Text Mining is. The efforts can be concluded defining Text Mining as the research area attempting to detect and extract meaningful information out of natural language text. This definition does not cover the necessary methodology to achieve this goal. Witten & Frank (2000) see Text Mining as the application of data mining methods to textual data. This may be considered to be true for some tasks (e. g. from the field of Information Retrieval), for which it suffices to represent a document as feature vector containing information about the presence or absence of words. This is not the case for more complex tasks demanding more sophisticated approaches. It is todays common understanding that Text Mining can be seen as a challenging task as indicated by Gao et al. (2005):

> Text mining is the art and technology of extracting information and knowledge from text collections [...].

Compared to structured data which is explored by data mining methods, textual data is unstructured, contains ambiguous words and is amorphous.

Data Mining is looking for patterns in data while Text Mining looks for patterns in text. Although both fields seem to be similar to each other, major differences appear when looking below the surface. Data Mining extracts implicit knowledge from the data that was previously unknown (see Witten & Frank, 2000). This information is hidden and not stated explicitly. Contrarily, the information extracted by Text Mining is explicitly uttered in the text. It is not

hidden and in most cases unambiguously. The challenge for Text Mining is the data representation itself. Textual data is not stored in a structured way like in databases, instead it is couched by natural language. Natural language is not suitable for machine learning approaches directly and thus, Text Mining strives to extract valuable information and to transform it into a more structured, machine-processible form.

Mehler & Wolff (2005) present four different perspectives from which to consider Text Mining: the Information Retrieval Perspective, the Data Mining Perspective, the Methods Perspective, and the Knowledge-oriented Perspective. Regarding the third perspective, Text Mining can be seen as a collection of methods for automatic processing of natural language data. Methods from various fields can be applied. *Information Retrieval methods* are used to extract relevant documents or to create topic-specific corpora. *Linguistic methods* are employed to augment unstructured textual data with structural information. This includes proper tokenization, lemmatization, part-of-speech tagging and syntactic parsing. Based on these linguistic enrichments, *Information Extraction methods* detect entities, proper entities and extract relations between different entities. *Statistical and Data Mining methods* are then applied to extract implicit knowledge based on extracted entities, relations and the like. Furthermore, *classification methods* are used to classify texts or parts of texts into topic-specific groups, while automatic *text summarization* is used to concentrate on the essential parts of the data.

According to Feldman & Sanger (2007) one of the major differences between Data Mining and Text Mining is the preprocessing step. While preprocessing for Data Mining concentrates on scrubbing and normalization of the data, Text Mining requires a comprehensively identification and extraction of natural language features to represent the information contained in the document. This step achieves a structured storage of unstructured data which is not necessary for most Data Mining systems as the data is already stored appropriately.

This thesis deals with this preprocessing step essential for every Text Mining system. It focusses on domain-specific Information Extraction out of textual data which relies on linguistic processing steps. Due to the unstructured format of

natural language data, linguistic processing is crucial to dive into the structure of textual data in order to generate structured features useable for application of Data Mining methods.

In the following, a short introduction to machine learning approaches employed in NLP is given in Sec. 4. As applications of these approaches are grounded on linguistic considerations, a brief overview of linguistic foundations is presented in Section 4.2. Finally, both automotive corpora are presented in Section 5 along with corpora used for evaluation purposes. Following this, the performance of state-of-the-art algorithms when being applied on domain-specific data from the automotive domain is analyzed in Section 6.

The second part of this thesis deals with unsupervised methods for Information Extraction. As the analysis will show, most approaches do not perform well on automotive data. Hence, unsupervised methods for NLP are developed to consider the specific linguistic properties of the data. These algorithms include part-of-speech tagging (see Section 7) and syntactic parsing (see Section 8). Finally, a statistical approach to RE is presented in Section 9 building the actual core algorithm for feature extraction in terms of relations.

3

INFORMATION EXTRACTION

The field of IE emerged during the last decades and bridges the gap between keyword based methods and highly sophisticated efforts towards complete language understanding.

Starting with a conference series initiated by the Defense Advanced Research Projects Agency (DARPA) – the *Message Understanding Conferences* (1987 – 1997, see Grishman (1995); Grishman & Sundheim (1995)) – information extraction became a clearly defined tasked focusing on the extraction of facts and relations between different entities. For each challenge, pre-annotated training and test sets were provided to the scientific community containing texts from real news corpora (e. g. the extraction of terrorist attacks or joint venture announcements). The importance of NER as independent subtask has first been underlined during MUC-6. Since the origins during the *Message Understanding Conferences*, several Named Entity types are well established. Those types contain amongst others: proper names of people, companies / organizations, places and numbers like date, time, money or percent values (see Borthwick et al., 1998; Sang, 2002; Sang & Meulder, 2003). The actual task of relation extraction was called template filling as it was formulated as a slot-filling task of templates.

The task of information extraction is divided into at least two successive subtasks:

ENTITY DETECTION Entities are the things the information and facts given in texts is about. Entity types cover a huge variety including the named entities mentioned before and basically every nominal phrase (e. g. noun

phrases like *the table* or *my car* and pronouns like *he*). Everything in a sentence that can fill a slot of a relation is an entity. Consequently, the task of entity detection is to detect all of the required entities.

RELATION EXTRACTION A very intuitive and vivid definition of relation was shaped by De Morgan (1966):

> When two objects, qualities, classes, or attributes, viewed together by the mind, are seen under some connexion, that connexion is called a relation.

From a formal point of view, a relation R over the entity sets $E_0, \ldots, E_n$ is a subset of their Cartesian product:

$$R \subseteq E_0 \times \ldots \times E_n \tag{3.1}$$

Additionally, each relation can have a type which defines the kind of the connection more precisely (e. g. *is-head-of* vs. *is-employee-of*). Following the spirit of both definitions, the task of Relation Extraction is to extract tuples of n entities that present an instance of a particular n-ary relation.

3.1 ENTITY RECOGNITION

Most approaches to entity recognition focus on named entities. Named entities are entities for which rigid designators (or *Kripkean soul*) stands for the referent (see Kripke, 1980). As defined in most NER tasks (see Sang, 2002), it is general agreement that temporal expressions and numerical expressions (e. g. money values, percentages, etc.) are also considered named entities. When aiming on extraction of domain-specific entities, entities beyond the scope of classical NER tasks have to be thought of (e. g. names of genes in molecular biology as in Settles (2004)). Regarding the automotive domain, vehicle-related entities (see Table 3, p. 71) and certain codes for parts, actions and quality assurance are of special interest. Most of these are represented by arbitrary noun phrases (e.

g. *My car*, *a puncture*) and thus, classical NER approaches do not recognize the entities in question.

Entity recognition can be divided into two parts: *entity detection* and *entity classification* (see Krauthammer & Nenadic, 2004). Entity detection is the task of marking single or adjacent words denoting an entity. Entity classification assigns the proper entity type to detected entities afterwards. Although several machine learning approaches exist for both subtasks (see Collins (1996) for proper noun tagging and Collins & Singer (1999) for entity classification), classification of the entity's type is often achieved as result of the training on annotated data (see Sang, 2002). Actually, all machine learning approaches can be employed (with different success and depending on feature selection, see Nadeau et al. (2006)) to entity recognition, but especially named entity detection can be achieved by simpler methods, e. g. POS Tagging.

Starting with the Message Understanding Conferences, rule based methods achieved the best results for many years (see Chieu & Ng, 2002). The advantage of rule based systems is the independence from manually annotated training data and thus, they are a suitable option for fast and flexible domain adaptation, although some effort has to be put into rule creation.

Since more and more annotated corpora became available to researches, approaches based on classifiers (see Isozaki & Kazawa, 2002; Kazama et al., 2002) achieve better results and lead to combined models for entity detection and classification. All of them are based on either manually created rules (see Appelt et al., 1992), dictionaries containing entity information (see Borthwick et al., 1998) or annotated training data (see Sang, 2002). As manual annotation of these resources is time-consuming and costly, often none the ones above are available for various languages and / or domains. Hence, it is a challenge to adapt ER to other domains or languages as it strongly depends on these resources, feature selection and the extend of available annotated corpora (see McCallum & Li, 2003).

In order to reduce the amount of manual annotation, semi-supervised methods are used to extend a small set of seed annotations. This strategy is called *boot-*

strapping (see Brin, 1998; Agichtein & Gravano, 2000; Kozareva, 2006). Bootstrapping methods are iterated (semi-)supervised classifiers that are initially trained on a small training set of annotated instances. In each iteration, new instances are generated – optionally filtered in a supervised fashion – and added to the seed set which grows steadily and results in a comprehensive training set.

Most supervised approaches can be turned into an iterated bootstrapping algorithm, although the original approach concentrates on extraction of patterns (see Riloff, 1996; Riloff & Thelen, 2002). Several criteria can be employed to decide when to stop the iteration, e. g. a certain number of iterations, convergence or further conditions (see Borthwick et al., 1998; Collins & Singer, 1999; Chieu & Ng, 2002).

Brin (1998) suggests a combined module for NER and RE, but using separate modules increases robustness and retains convenient modularization.

3.2 RELATION TYPES

Relation extraction can deal with several types of relations. However, sophisticated extraction algorithms rely strongly on relation characteristics, such as arity, entity and relations types.

Simple *semantic relations* are given by synonymy and antonymy, hyponymy (*hyponym* is-a *hyperonym*), meronymy (*meronym* is part of *complex thing*) and co-hyponymy (*co-hyponym a* and *co-hyponym b* are *hyperonyms*). Those relations are given in thesauri such as WordNet[1] (see Miller et al., 1990) or GermaNet[2] (see Hamp & Feldweg, 1997) that often serve as knowledge base for relation extraction systems (for enrichment with hypernyms see Culotta & Sorensen, 2004). These relations can be extracted using corpus statistics (see Biemann & Bordag, 2003; Mahn, 2005) or pattern based approaches (see Hearst, 1992). Especially

1 http://wordnet.princeton.edu/
2 http://www.sfs.uni-tuebingen.de/lsd/

pattern based approaches relying on lexico-syntactic patterns (e. g. *NP like other NP* or *NP is a NP*) achieve very accurate results.

Although semantic relations can be rather arbitrary with regard to arity, complexity, the use of nested relations and involved entity types, the scientific community focuses on *linguistic relations* and binary logical relations. Linguistic relations (e. g. syntactic relations, dependency relations, head-modifier relations) are dealt with in Section 8, this section focusses on (binary) logical relations.

Logical relations yield informative semantic knowledge about arbitrary concepts and entities. Some relations are well established in the scientific field for comparison of different approaches. Examples are mergers of companies (see Yangarber et al., 2000), companies and associated locations of their headquarters (see Culotta & Sorensen, 2004) or relations between authors and books (see Brin, 1998). These relations are widely established and often reconsidered in works on RE due to easier comparison with other approaches.

Regarding the automotive domain, logical relations are the ones to be extracted. Table 4 (p. 72) gives an overview about the most important automotive relations.

3.3 RELATION IDENTIFICATION

Before applying any methodology to extract relations, target relations have to be identified in either supervised or unsupervised manner. Relations between entities are characterized by the entities and their types along with the type of the relation. The type of a relation specifies the type of connection between the corresponding entities. For most constellations, more than one possible type is applicable. Even different relations between identical entities exist. Regarding the entities PERSON *Will Smith* and MOVIE *Hancock*, at least two relation types can be annotated: *is-actor-of* and *is-producer-of*[3].

3 see http://www.imdb.com/title/tt0448157/

In most cases, relations are manually defined and a huge amount of human effort has to be put into adaptation to new domains and scenarios. Starting with the Message Understanding Conferences, certain entities and relation types were established and almost every algorithm is trained and evaluated on those types. Most tasks provide an annotated data set and with it, entities and relations.

In order to reduce the manual effort, Shinyama & Sekine (2006) propose an approach to *Unrestricted Relation Discovery*. The basic idea is to cluster mentions of entity pairs to detect different relation types. This *preemptive* processing step generates instances of all extracted relation types and thus, can be used to annotate training data for relation extraction methods. It only relies on entity annotations, but as entity detection is a highly domain-dependent task, the most suitable entity recognition approach (regarding the task's context) can be used.

Unsupervised Relation Identification (URI) – a similar approach based on Unrestricted Relation Discovery – is presented by Rosenfeld & Feldman (2007). The task of URI is defined as "an automatic discovery of all interesting relations in a large body of text" (see Rosenfeld & Feldman, 2007). A relation can be descriptively defined by a representative set of instances. The task of URI is to extract relation types along with such a representative set of instances.

This descriptive definition of relation type enables fully unsupervised relation extraction provided that entity annotations exist. Supervised approaches do not rely on any kind of relation identification as they automatically learn the correct type given in the training data.

3.4 RELATION EXTRACTION

The task of RE is defined as the extraction of instances for a given set of target relation types. These relations belong to the class of logical relations for which instances contain a type and the corresponding related entities.

Prerequisites of RE are entity recognition and relation identification, both can be achieved in a supervised or unsupervised manner.

Relation Extraction approaches can be divided into two classes – pattern-based approaches and kernel-based approaches. All approaches need training data containing instances of the target relations. Although they differ in number of necessary instances, at least some examples have to be annotated manually.

Many approaches to unsupervised Relation Extraction exist (e. g. Rosenfeld & Feldman, 2006b; Feldman & Rosenfeld, 2006; Yan et al., 2009), but nearly all of them rely on knowledge that was extracted in a supervised or semi-supervised preprocessing step. In most scenarios, named entity sets and / or relations are considered to be given.

3.4.1 Pattern-based approaches

Pattern-based relation extraction is achieved by mainly four steps (see Auger & Barrière, 2008):

DEFINING TARGET RELATIONS Various relation types stand in the focus of Information Extraction. While simple semantic relations like hypernymy (see Caraballo, 1999), meronymy (see Winston et al., 1987; Berland & Charniak, 1999; Girju et al., 2003; Pennacchiotti et al., 2006) and definitional relations (see Pasca, 2005) are studied thoroughly, some logical relations also gain a lot of attention (*located-in*, *book-authored-by*, *birthdate-of* as in Blohm & Cimiano (2007); Ravichandran & Hovy (2002)). Pantel et al. (2006) studied an extraordinary set of relations containing succession (e. g. for presidents: Bush and Obama), chemical reactions (e. g. magnesium and oxygen) and production (e. g. hydrogen and metal hybrids) showing applicability of pattern-based approaches to both factual knowledge extraction and encyclopedic information extraction.

DISCOVERING PATTERNS EXPRESSING TARGET RELATIONS Research may have two opposite points of view – an onomasiological and a semasiological one. Most work on RE follow the onomasiological approach using a predefined set of target relations and extract patterns expressing these relations as

precisely as possible. Some work use the opposite approach. A relation identification step is integrated to automatically detect clusters containing relation instances, each referring to a certain relation type (see Shinyama & Sekine, 2006; Rosenfeld & Feldman, 2006a, 2007). For the latter approach, each relation instance cluster gets a proper relation type assigned, which is described by the contents of the cluster.

The major strategy of pattern-based approaches is to use the contextual lexical units of an instance to compile a list of reliable patterns for each relation type. Analogously to entity extraction, this can be turned into an iterative bootstrapping method to increase coverage of the target relations (see Hearst, 1992; Brin, 1998).

Although most approaches initially start with a few entity pairs instantiating a relation, some recent work uses a seed set consisting of reliable patterns (see Etzioni et al., 2004) or keywords (see Rosenfeld & Feldman, 2006b). It is noteworthy that all bootstrapping strategies to semantic relation extraction (as for most other tasks) require any kind of quality control for newly learned patterns to avoid drifting (see Auger & Barrière, 2008).

Even in extensive training corpora, certain relations appear only in a limited number of variations and data sparseness becomes a problem. Cimiano et al. (2003a) criticizes pattern-based approaches in that manner:

> The approaches of Hearst and others are characterized by a (relatively) high precision in the sense that the quality of the learned relations is high. However, these approaches suffer from a very low recall which is due to the fact that the patterns are very rare in corpora.

SEARCHING FOR NEW INSTANCES Given some patterns for a relation, instances of that relation can be detected and the corresponding entities can be assigned to their respective roles. Regarding a basic example: instances of the relation *is-a(X, Y)* can be detected matching the pattern *x is a y* in a text.

The mentions of x and y will be assigned to the relation's roles X and Y, respectively.

Measuring confidence of extracted patterns is often achieved by the pattern's pure frequency. Another important aspect is the *specificity* of a pattern – the capability at expressing the relation in question only. Specificity is thoroughly explored in Alfonseca et al. (2006) and Turney (2006) additionally defines the *pertinence* of a pattern with respect to a specific entity tuple. Although a lot of effort has been put into evaluation of extracted patterns, there is still space for improvement. This evaluation step is crucial for any approach based on bootstrapping as it controls the quality of the method decisively.

STRUCTURING NEW INSTANCES Detected relation instances represent facts and extracted knowledge. In order to make access to it feasible, the resulting instances need to be stored in a formalized and structured way. Several standards (e. g. Resource Description Framework (RDF) and Web Ontology Language (OWL)) exist for formalization and structuring conceptual classes along with instances of those. These standards can be implemented creating ontological resources being capable of storing extracted facts and knowledge.

3.4.2 Kernel-based approaches

Recent research on machine learning methods for RE focuses on sophisticated kernels and Support Vector Machine (SVM)s. As most supervised machine learning algorithms, positive and negative instances are necessary to train a SVM. Relation instances need syntactic enrichment (e. g. part-of-speech tags as in Collins & Duffy (2001), dependency trees as in Culotta & Sorensen (2004), constituency trees as in Moschitti (2004), entity and role annotations as in Zelenko et al. (2002, 2003)) to be valid training examples. After obtaining a trained model, new sentences can be classified. Multiple resulting classifications are possible –

a sentence may contain a relation, various relations of different types or no relation at all. Some decisions influence the performance of such a classifier, some of these decisions are:

- employ multiple binary classifiers or one multi-class classifier[4],
- augment data with dependency parse trees or constituency parse trees[5], and
- selection of the kernel function[6].

Some researches argue in favor of dependency parsers (see Fundel et al., 2006; Wang et al., 2006), but Jiang & Zhai (2007) testify inferior performance of dependency parses in the task of RE. The choice of the kernel function mostly influences the results. Instances are not represented as feature sets like in other machine learning approaches. Instead, the kernel function explores the instance on a complex object level without any transformation.

The annotation of the training set is easier than for most other tasks. The augmentation with syntactic and semantic information can be done in an automatic way. The only manual effort is the annotation of present relation types.

Employing this approach for RE reduces the extraction of relation instances to a simple classification task.

4 see Section 4.5 for more details
5 see Section 4.2.4 for differences among grammar theories
6 see Section 4.6 for suitable kernel functions

4

MACHINE LEARNING ALGORITHMS

Machine learning algorithms are employed to turn data into knowledge. Assuming sufficient amounts of available data, machine learning algorithms try to extract information, detect patterns or to infer knowledge that is (implicitly) hidden in the data.

From a psychological perspective, machine learning is of special interest as it can facilitate understanding human learning itself. Formalization of learning theories forces researches to go into detail of every aspect of the theory. Comparing the performance of the model with humans provide insights about shortcomings of the theory and thus, can suppose reasonable steps towards improved models.

Most approaches to machine learning apply *supervised* methods. Training examples containing input to the algorithm along with the desired output are used to create or iteratively improve a model. This approach can be compared to the learning of children which are exposed to samples provided by their parents and other people. Especially language acquisition by children is a field of rising interest (see Schütze & Walsh, 2008) and appropriate data sets consisting of communication between children and their playmates, siblings and caretakers is freely available (see MacWhinney, 2000).

Regarding NLP, various machine learning techniques operating on different levels of supervision (see Sec. 4.1) are employed for different tasks. Among the most important ones are *naive Bayes classifiers* based on *Bayes' Theorem* (see Bayes & Price, 1763), *Neural Networks* inspired by human brains and *Decision Trees* (see Pagallo, 1990).

Three approaches will be explained in detail as they are often employed for the tasks of part-of-speech tagging (Hidden Markov Models, see Sec. 4.3; Clustering Analysis, see Sec. 4.4), syntactic parsing (Clustering Analysis) and Relation Extraction (Support Vector Machines, see Sec. 4.5; Kernels, see Sec. 4.6).

4.1 LEVELS OF SUPERVISION

The level of supervision denotes the amount of human effort that has to be put into training of an algorithm. Normally, three different levels of supervision are distinguished:

SUPERVISED ALGORITHMS Supervised algorithms are trained on training samples. Training samples consist of the *input* that is presented to the algorithm and the desired *output* which is expected to be generated by the algorithm. Training with a given training set is often performed iteratively until the model of the algorithm converges so that the algorithm produces the expected results. Supervised methods proved to achieve very accurate results for NLP tasks under the condition that sufficient training samples are available. The drawback of supervised methods is the creation of such training samples which needs to be done manually in most cases. Furthermore, the annotation of a large amount of data is necessary for each language, domain and text type.

SEMI-SUPERVISED ALGORITHMS Semi-supervision relies on large amounts of unlabeled data and a small seed set of annotated data. The seed set is used to initially train a model which is then used to annotate the unlabeled data. The union of the newly annotated instances and the initial seed set is used afterwards to train an improved model. This is done iteratively until previously defined abort criteria (e. g. convergence of the model, maximum number of iterations) are met. Additional human supervision can be ap-

plied to validate the newly extracted instances to keep error propagation to a minimum and ensure accurate results.

UNSUPERVISED ALGORITHMS Completely unsupervised algorithms solely rely on unlabeled data. This leads to greatest independence from human effort and consequently, unsupervised algorithms can be easily applied to new data sets. As unsupervised algorithms cannot use human annotations, extracted knowledge is unlabeled, too. E. g. it is possible to cluster data like documents or words inducing clusters containing documents about a distinct topic or words belonging to the same part-of-speech. The resulting clusters cannot be labeled by topic x or part-of-speech y as no information about the topic or correct part-of-speech is given. Instead, descriptive labels can be defined for each cluster using common properties of its members (e. g. word cluster containing *sunday, car, lamp, etc.*).

4.2 LINGUISTIC FOUNDATIONS

Most approaches to Information Extraction rely on linguistic features. In this section, a short introduction to the foundations of corpus linguistics is given to finally present machine learning approaches exploiting these features.

Starting with de Saussure (see de Saussure, 1966), linguists shifted from diachronic to synchronic analyses describing language as interconnected lexical units of different levels. In modern corpus linguistics, utterances of a language are collected to build a *corpus* which is then thoroughly explored. Exploration of language data includes classification of elements of different linguistic levels (see Sec. 4.2.1). Furthermore, de Saussure proposed to study structural phenomena like collocations between units. Current approaches to syntax and semantics employing co-occurrences (see Sec. 4.2.2.2) are based on syntagmatic and paradigmatic analyses (see Sec. 4.2.2) which were originally proposed by de Saussure (see Heyer & Bordag, 2007).

With huge corpora being available, corpus linguistics shift to employ distributional approaches describing units through distributional properties. Especially classification into linguistic categories is modeled by context distributions (see Sec. 4.2.3). These ideas go back to Harris (1954) and Firth (1957) who initially state that similar words occur in similar contexts or are paradigmatically related to use de Saussure's terminology. In more recent research, even more complex syntactic phenomena such as grammars (see Sec. 4.2.4) can be explored by distributional approaches (see Klein & Manning, 2002b).

4.2.1 Levels of language

Language is obviously structured in a hierarchical way. Humans know the difference between *diesel* and *diesel engine*. Although the pure character strings show a significant similarity at their beginnings, the entities denoted by these two strings are completely different ones. Furthermore, it seems that the shorter string (*diesel*) is a part of the second one which has been built of it and another entity (*engine*) following some rules.

Two composition rules exist to form complex linguistic elements:

COMPOSITION Composition of elements aggregate elements to a complex one. This composition underlies certain language-dependent rules which have to be obeyed in order to produce a linguistic unit of the next level. Multiple units cannot be uttered at the same time and thus, the order of involved lexical units is important to maintain the intended meaning. Even if some rules are violated, human language processing might be able to understand the utterance.

ABSTRACTION Atoms of a language level can be classified into equivalence clusters regarding to some of their properties. This is called abstraction or selection. The members of such a cluster show similarity for some properties and can be distinguished by other properties. Abstraction is a very

commonly employed method throughout language processing and will be employed for extraction of various relation types.

Linguists distinguish between several levels. A very common hierarchical structure is based on Harris (1968):

PHONEMES / GRAPHEMES The basic atoms of spoken language are *phones*. Multiple phones build *phonemes* which are the smallest units distinguishing meaning of an utterance. Usage of phonemes is not language independent. Some phonemes are used in a language very commonly and barely in another. *Graphemes* are the written forms of phonemes. The transformation of phonemes into graphemes is not unambiguous and for different languages using different grapheme sets (called *alphabet*, e. g. Latin alphabet: a, b, c, ...; Greek alphabet: α, β, γ, ...) identical phonemes may be encoded through different graphemes.

MORPHEMES *Morphemes* consist of one or more graphemes and form the smallest units bearing meaning. Two different types of morphemes can be distinguished: free and bound morphemes. *Free morphemes* (e. g. *dream*) can build words by themselves. *Bound morphemes* (e. g. *ing* as in *dream-ing*) are not allowed to occur without a free morpheme as they only bear syntactic function. It is common for some languages (e. g. German), that multiple bound morphemes are attached to a stem.

WORDS A valid combination of morphemes is called *word form*. The class of semantically equivalent word forms is denoted by *word*. A word is represented by its basic form – nominative singular for nouns and infinitive presence for verbs, respectively. E. g. the word *dream* includes at least the word forms *dream*, *dreaming*, *dreamed* and *dreams*.

PHRASES / CONSTITUENTS *Phrases* consist of morpho-syntactic combinations of word forms. All phrases are *constituents* which build the basic elements of a sentence. Within a constituent, word forms have to show agreement on certain grammatical properties. For nouns, these properties are: person,

number, gender and case (noun only). Verbs additionally need to agree on tense, aspect, mood and voice. A sequence of words can be tested for constituency employing so-called constituency tests (see Adger, 2003).

SENTENCES A valid (according to grammatical constraints) hierarchical structure built of constituents is denoted as well-formed *sentence*. Complete sentences are usually uttered in human communication – including short ones leaving parts out (e. g. ellipsis).

TEXT / DOCUMENT Larger elements like texts (often referred to as documents in the context of Information Extraction) consist of multiple sentences and optional meta-information (e. g. author, date).

CORPUS A corpus is a collection of textual data. It is often enriched with linguistic annotations of any level – tokens, parts-of-speech, phrases / parse trees.

4.2.2 Syntagmatic and Paradigmatic Relations

Syntagmatic and paradigmatic relations are the two fundamental relation types for lexical items (see Heyer et al., 2006). According to de Saussure (1966), two lexical items stand in *syntagmatic* relation if they occur together. Similarly, a *paradigmatic* relation between two units exists, if they are observed within similar contexts.

In order to use these two definitions for computational language processing, they need to be expressed in a formal way. Let $\mathcal{L} = \{W, S\}$ be a language with a set of all of its word forms W (word forms contained in W are denoted by w) and all of its sentences S (sentences contained in S are denoted by s).

Each sentence s (with length n) is defined as an ordered sequence consisting of word forms:

$$s = \langle w_1, \ldots, w_n \rangle \tag{4.1}$$

4.2.2.1 *Context*

The *local context* $\mathrm{context}(w, s)$ of a word w in a sentence s consists of words that can be observed within a given distance of w. The distance can be defined rather arbitrarily, but certain definitions are well established as they proved to achieve reasonable results for different tasks.

Among the most prominent contexts is the *neighborhood context*. It usually consists of the preceding and succeeding words of the target word.

More formally:

$$\mathrm{context}_{\mathrm{NB}(d)}(w_x, s) = \langle w_{x-d}, \ldots, w_{x-1}, w_{x+1}, \ldots, w_{x+d} \rangle \text{ with } w_i \in s \tag{4.2}$$

Neighborhood context must not exceed sentence boundaries.

Choosing context size d big enough leads to a context containing all words of the sentence: the *sentence context*[1]:

$$\mathrm{context}_S(w_x, s) = s \backslash w_x \tag{4.3}$$

The *global context* $\mathrm{context}(w_x)$ is the sum of all observed local contexts of a given word w_x with respect to a corpus.

Both local and global contexts can be altered using appropriate filters. The choice of a filter strongly depends on the task and influences the performance decisively. Common filters discard low frequent occurrences of context entries (e. g. for word clustering) or entries regarding to their part-of-speech (e. g. for synonym extraction).

4.2.2.2 *Syntagmatic Relations*

Using the definition of local context, two word forms $w_x, w_y \in W$ are syntagmatically related if and only if a local context exists that contains both word forms. The occurrence of two word forms in a certain context is called *co-occurrence*.

$$\mathrm{SYN}(w_x, w_y) \leftrightarrow \exists s \in S : w_y \in \mathrm{context}(w_x, s) \tag{4.4}$$

1 Contrary to the neighborhood context, word order is ignored for sentence context calculation.

The construction of natural language sentences is nearly unrestricted (except for rules ensuring grammaticality) and thus, nearly every word form could co-occur with all other word forms in a sentence. This results in syntagmatic relations between all possible word form pairs. The actual question is: are there any pairs of word forms standing in a statistically significant syntagmatic relation?

To decide which word form pairs occur in a statistically significant relation, a significance measure is applied (see Sec. 4.2.2.4).

Consequently, two words $w_x, w_y \in W$ stand in a statistically significant syntagmatic relation $SYN_{sig}(w_x, w_y)$ if they are syntagmatically related and their co-occurrence is statistically significant with respect to a significance measure (see Heyer & Bordag, 2007).

The choice of context calculation and the applied significance measure influences the results. Significant co-occurrences can be used to study various linguistic phenomena like dependencies (e. g. *sun shines*), idioms (e. g. *make your day*), syntactic relations (e. g. *the sun*) and multi-word units (e. g. *The Ney York Times*).

4.2.2.3 *Paradigmatic Relations*

Two words stand in paradigmatic relation $PARA(w_x, w_y)$ if and only if their respective global contexts are similar with respect to a similarity measure SIM (see Sec. 4.2.2.5):

$$PARA(w_x, w_y) \leftrightarrow SIM(context(w_x), context(w_y)) \tag{4.5}$$

While syntagmatic relatedness is derived from local properties, paradigmatic relatedness is a global property depending on all sentences of a given language. It is common sense to use filters (e. g. significance filter and threshold to thin out the local contexts, part-of-speech filter to allow only words of the same part-of-speech) to improve results.

Words occurring within similar contexts – and thus, standing in paradigmatic relation – often belong to the same *semantic cluster*. The words *sun*, *lamp* and *bulb* all share the property to shine and thus, occur within similar sentence contexts.

They also occur within similar neighborhood contexts as they share the part-of-speech property.

4.2.2.4 *Significance measures*

Significance measures are employed to decide which co-occurrences are statistically significant and which are not. Several measures have been proposed, most of them are based on either bigram probabilities (e. g. Frequency Measure), statistical independence (e. g. Mutual Information) or likelihood measures (e. g. log-likelihood).

A comprehensive study on significance measures is given in Pecina (2005). Depending on the task, different measures fulfill the requirements of the objective in different quality levels. A combination of multiple measures may improve the overall performance significantly as properties of the measures can be combined with respect to the task (see Pecina & Schlesinger, 2006). Computation of co-occurrences and their significance values is a complex task for big corpora. A fast and efficient solution to this problem is presented in Büchler (2006).

In the following sections, the principles of some of the most prominent measures are presented. All measures are defined to calculate the significance of the co-occurrence of the words A and B where n_a denotes the frequency of word A in the given corpus with size n, n_b is analogously defined as $n_b = f(B)$ and n_{ab} denotes the co-occurrence AB.

FREQUENCY MEASURE The most basic measure is provided by pure frequency:

$$sig_{freq}(n, n_a, n_b, n_{ab}) = n_{ab} \tag{4.6}$$

The frequency measure is used to extract co-occurrences that occur more often than a defined threshold n_{min}. Due to independence from n_a, n_b and n, co-occurrences that would intuitively be classified as significant (e. g. $n_a = n_b = n_{ab}$) achieve low scores compared to co-occurrences showing a higher frequency (e. g. $n_{cd} > n_{ab}$ with $n_c, n_d >> n_{cd}$).

DICE COEFFICIENT The Dice Coefficient originally was proposed to calculate similarity of sets (see Dice, 1945) and can be derived from the harmonic mean:

$$\text{sig}_{\text{dice}}(n, n_a, n_b, n_{ab}) = \frac{2n_{ab}}{n_a + n_b} \tag{4.7}$$

The resulting significance is normalized into the range $[0 \ldots 1]$ and additionally uses the frequency of the words. This measure shows interesting properties: the most significant co-occurrences according to this measure consist of words with $n_a \approx n_b$ and the most insignificant co-occurrences consist of a high-frequency and a low-frequency word (see Büchler, 2006).

MUTUAL INFORMATION Mutual information is based on statistical independence. It is defined as the ratio of actually observed frequency n_{ab} and expected frequency under the assumption of statistical independence. The logarithm is employed for numerical reasons. The probabilities can be very small and thus, multiplying them is computationally difficult.

$$\begin{aligned} \text{sig}_{\text{MI}}(n, n_a, n_b, n_{ab}) &= \log_2 P(n_{ab}) - \log_2 P(n_a) P(n_b) \\ &= \log_2 \frac{P(n_{ab})}{P(n_a) P(n_b)} \end{aligned} \tag{4.8}$$

Mutual information can also be derived from an entropy-based approach (see Büchler, 2006).

Low frequency co-occurrences obtain high scores using Mutual Information. To circumvent the low ranking of high frequency co-occurrences, a combined measure is proposed by Evert (2004) namely local mutual information:

$$\begin{aligned} \text{sig}_{\text{LMI}}(n, n_a, n_b, n_{ab}) &= \text{sig}_{\text{freq}}(n, n_a, n_b, n_{ab}) \, \text{sig}_{\text{MI}}(n, n_a, n_b, n_{ab}) \\ &= n_{ab} \log_2 \frac{P(n_{ab})}{P(n_a) P(n_b)} \end{aligned} \tag{4.9}$$

With the additional factor, high frequency co-occurrences will be ranked higher than on pure mutual information.

LOG-LIKELIHOOD Log-likelihood is based on binomial distribution and statistical independence (see Dunning, 1993).

(4.10)
$$\text{sig}_{\log}(n, n_a, n_b, n_{ab}) = \binom{n}{n_{ab}} p^{n_{ab}} (1-p)^{n-n_{ab}} \text{ with } p = p(n_a)\, p(n_b)$$

For this measure, n denotes the number of sentences and as property of the binomial distribution, the restriction $n_{ab} \leqslant n$ has to be hold. Analogously to the mutual information measure, the log-likelihood ratio measures the deviation of the actual observed frequency from the expected frequency of a co-occurrence.

4.2.2.5 *Similarity Measures*

Similarity measures are essential to compare things with each other. Regarding NLP tasks, feature vectors often need to be compared (e. g. tests for paradigmatic relatedness which need to compare global contexts of words). Global contexts contain all local contexts and are treated as vectors for easier computation.

As all vectors can be represented as sets, suitable similarity measures can be either defined on sets or directly on vectors. In the following, two prominent similarity measures for similarity calculation of two feature vectors / sets A and B are presented.

COSINE MEASURE Cosine similarity is probably the most established similarity measure. It measures the cosine of the angle θ between two feature vectors. This similarity measure takes values in $[0 \ldots 1]$ as the cosine of 0 is 1 for perfect agreement and is lower for all other values.

(4.11) $$\text{sim}_{\cos}(A, B) = \cos(\theta) = \frac{A \cdot B}{|A|\,|B|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

The cosine measure is often applied to document comparison (see Wilkinson & Hingston, 1991) or to measure cluster cohesion (see Tan et al., 2005).

JACCARD COEFFICIENT The Jaccard coefficient was originally proposed by Jaccard (1901). It is used to compare similarity and diversity of two sets A

and B. The Jaccard index denotes the ratio of the intersection's size and the size of the union of both sets as given in:

$$\text{sim}_{\text{Jaccard}}(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{4.12}$$

4.2.3 Linguistic Categories

One of the most fundamental linguistic distinctions are provided by *linguistic categories*. Elements of all language levels can be assigned to certain categories, distinguished by syntactic features into part-of-speech classes on the word level, into vocals and consonants one the phonem level, into various phrase types on the constituent level to name just a few possibilities.

Depending on the language level, the variety of classes and degree of differentiation differs significantly. Regarding the word level, words can be distinguished by syntactic or semantic properties. A basic distinction of syntactic functions can be achieved by classification of words into the four main categories *nouns*, *verbs*, *adjectives* and *function words* (see Bordag, 2007). Although this categorization seem to suffice for some tasks (e. g. ER as in Schierle (2011)), more fine-grained classifications are employed for most linguistic tasks.

A set of part-of-speech classes is referred to as tagset. While the classification into four main classes can be applied to all languages, some classes are only employed for a distinct language. Thus, different tagsets were proposed, each for a certain language. Due to differences in languages regarding syntactic complexity, corresponding tagsets differ in size and granularity. The *Penn Tree Tagset* (see Marcus et al., 1993) is a well-established tagset for the English language. It contains 45 tags – 36 for words and 9 for punctuation (see Appendix A.1). The Stuttgart-Tübingen Tagset is a tagset for German (see Appendix A.2). It contains 54 classes – 51 for words and 3 for punctuation. Both tagsets differ heavily in degree and classes of distinction.

Both tagsets contain more than four classes, which is necessary for deeper linguistic studies. The distinction between normal nouns and proper nouns,

different verb classes (e. g. for modal verbs and different tenses) makes sense and facilitates further analyses like parsing.

Words belonging to the same syntactic class stand in paradigmatic relations. Thus, literature often speaks of distributional classes. In order to derive syntactic classes containing words with similar syntactic functions (e. g. such as nouns, proper nouns, adjectives, etc.), the neighborhood context is used to create global contexts (see Schütze, 1995). Paradigmatic relatedness can also be exploited to create semantic clusters. For this purpose, complete sentences build the respective local contexts. When using an appropriate significance measure, words occurring in similar contexts share semantic properties (e. g. *sun*, *candle*, *shines*, *bright*). As stated before, filters can be applied to retain only words of the same part-of-speech in each class (e. g. restriction to nouns will lead to *sun* and *candle*).

Distributional classes exist on all levels. Starting on morphemes bearing similar inflectional functions and ending on phrase types (e. g. *noun phrases*, *prepositionl phrases*). On text or document level, documents can be clustered regarding to their respective topics or text types.

4.2.4 Grammar Theories

The grammar of a language is a set of syntactic or structural rules. These rules have to be obeyed in order to create valid words, phrases and sentences of this language. The term grammar also denotes the scientific fields studying those rules, including phonology, phonetics, morphology, syntax, semantics and pragmatics.

Regarding relation extraction, two grammar theories are of special importance: Phrase structure grammar (PSG) (see Sec. 4.2.4.3) and Dependency Grammar (DG) (see Sec. 4.2.4.4). Syntactic trees of both theories are used in recent works on RE. In this section, foundations of formal languages are presented (see Sec. 4.2.4.1) along with a brief introduction to both grammar theories.

4.2.4.1 *Formal Languages and Grammars*

Formal grammars were introduced in the first half of the twentieth century by amongst others Post (1943), although this work benefited from earlier work (e. g. from Thue, see Thomas, 2010). The study of natural languages and the formal description of these did first begin with Chomsky (1956).

Each formal language relies on an alphabet Σ which is a finite nonempty set of symbols or terminals. A word w over Σ is defined as a finite sequence of symbols of the alphabet with length $|w|$. For each language, exactly one empty word ϵ exists with $|\epsilon| = 0$.

Concatenation of two words $a = a_1 \dots a_n$ and $b = b_1 \dots b_n$ will create a new word $ab = a_1 \dots a_n b_1 \dots b_n$. Naturally, concatenation of a word a and the empty word ϵ does not create a new word, instead results in $a = a\epsilon = \epsilon a$. a^n denotes the sequential concatenation of $n \geqslant 0$ copies of word a.

A *language* over an alphabet Σ is a set of words. The set of all possible words over Σ is denoted by Σ^* and the set of all nonempty words is denoted by Σ^+.

Based on these formal basics, a *grammar* can be defined as a quadruple (Σ, V, S, P), where:

- Σ is the terminal alphabet.
- V is a finite nonempty set containing nonterminal symbols. V is disjoint from Σ.
- S is a distinguished nonterminal symbol referred to as start symbol.
- P is a finite set of production rules of the form $\alpha \to \beta$ where α consists of terminals and at least one nonterminal and β consists of any number of terminal and nonterminal symbols.

4.2.4.2 *The Chomsky Hierarchy*

Constraints on the set of production rules can restrict the complexity and the production mightiness of a grammar. Chomsky proposed a grammar classification

into four classes (see Chomsky, 1956, 1963). Starting with an unrestricted grammar, restrictions increase with each class shaping this categorization's name: *Chomsky hierarchy*.

Let G be a grammar (Σ, V, S, P):

TYPE 0: UNRESTRICTED GRAMMAR G is an *unrestricted grammar* and no restrictions have to be satisfied.

TYPE 1: CONTEXT-SENSITIVE GRAMMAR G is a *context-sensitive grammar* if only non-shortening production rules exist, formally: $|\alpha| \leqslant |\beta|$. One exception to this constraint exists: the production rule $S \to \epsilon$ is allowed, if S does not occur on the right side of any rule.

TYPE 2: CONTEXT-FREE GRAMMAR G is a *context-free grammar* if the left side of all production rules satisfies $|\alpha| = 1$ or in other words: all left sides must consist of exactly one non-terminal symbol.

TYPE 3: REGULAR GRAMMAR G is a *regular grammar* if all production rules have one of the three types $A \to cB$, $A \to c$ or $A \to \epsilon$ where A and B are non-terminals ($A = B$ is allowed) and c is a terminal symbol. Grammars of this type are also called *right-linear* grammars as productions only operate on the right-most end of a production.

4.2.4.3 *Phrase Structure Grammar*

The term *Phrase Structure Grammar* denotes grammars consisting of *re-write rules* structuring a sentence based on hierarchical constituents. Phrase structure rules were originally introduced by Chomsky (1957), although this kind of rules have been studied previously by Thue (1914); Post (1943).

Linguists formalize the grammar by generative rules that assign constituency tree structures to sentences. Hence, PSGs are also called *constituency grammars*. The phrase structure rules are a consolidation of traditional subject-predicate structure and immediate constituent analysis of Bloomfield (1933). Constituency relations are defined by grammar rules that function in deep structure making explicit the domination of constituents over other constituents.

The phrase structure is usually represented as a parse tree or bracketing. The nodes of a parse tree are labeled by their respective constituent types (e. g. *N* for noun, *NP* for noun phrase, *VP* for verb phrase, etc.). A constituent *dominates* another one if it is placed directly above the other. Constituents can dominate multiple other constituents and thus, larger constituents can consist of multiple smaller ones. Thereby, the linear order of words and constituents is retained. A constituent *precedes* another one if it stands directly in front of the second constituent.

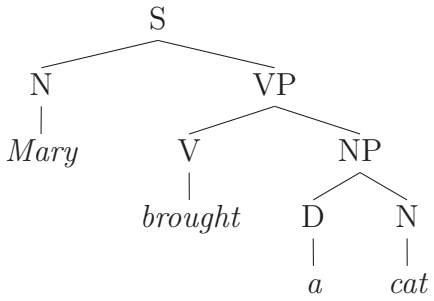


Figure 1.: Constituency parse tree for *Mary brought a cat*

The parse tree of Fig. 1 can be generated by six grammar rules:

S	→	N VP	D	→	a
VP	→	V NP	N	→	Mary, cat
NP	→	D N	V	→	brought

Table 1.: Grammar Rules for generation of *Mary brought a cat*

Phrase structure grammars are very strong in structural descriptions of a language. Despite its power in segmentation and categorization, yet it is incapable of accounting for all phenomena of natural languages. PSGs show difficulties in analyzing questions, ambiguities, discontiguous constituents and remote relationships.

Due to the deficient analyses in those cases and as reaction to the *Transformational Grammar* (see Chomsky, 1956), researchers put some effort into advancements of the PSG. Gazdar et al. (1985) introduce the *Generalized Phrase Structure Grammar* which aims on context-free description of natural language. Further-

more, syntactic derivations are augmented with semantic annotations and *meta-rules* to compute the meaning of a sentence. Most of the ideas were integrated in the highly lexicalized *Head-driven Phrase Structure Grammar* (see Pollard & Sag, 1994).

All kinds of PSG are based on constituents. Constituents are contiguous sequences of words that can be treated as a single lexical unit. Even dependency grammars (see Sec. 4.2.4.4) allow analysis of constituent parts.

Basically, each sequence of words is a candidate for being a constituent. In order to identify those sequences that function as a single unit, several constituency tests (see Radford, 1988) can be applied. These tests are not perfectly reliable and thus, a single test is not enough to make a decision for a candidate. To be sure about a sequence, as many tests as possible have to be employed. Each test applies a transformation to the sentence. If the sentence is still grammatically sound then the candidate could be a constituent.

The most important tests are:

FRONTING / TOPICALISATION TEST The candidate sequence is moved to the beginning of the sentence.

CLEFTING TEST The test sequence c is placed into the structure *It was c that* ....

REPLACEMENT TEST The candidate sequence is replaced by an appropriate pro-form (e. g. pronoun).

ANSWER ELLIPSIS TEST A question is created for which the test sequence is the answer.

PASSIVIZATION TEST An active sentence is transformed into a passive sentence, or vice versa. Hence, the object is changed to the subject and the other way around.

DELETION TEST The test sequence is omitted.

COORDINATION TEST The candidate sentence is coordinate with a constituent of the same type.

4.2.4.4 *Dependency Grammar*

While PSGs study constituency relations of a language, DGs are defined on *dependency relations*. Modern dependency grammars are inspired by Tesnière (1959) who first developed a formal description of dependency grammars. A *dependency* is a binary directed relation between two words, one of the words (the *dependent*) depends on the other one (the *head*). The major assumption of DGs is: All words of a sentence – except for one – depend on other words. Dependencies between words are either motivated syntactically or semantically. Words that are complements or modifiers of others depend on the latter and thus, multiple words can depend on the same head (e. g. transitive verbs require two complements). A dependency tree is given in Fig. 2:

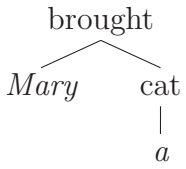


Figure 2.: Dependency tree for *Mary brought a cat*

Robinson (1970) formulated four axioms to ensure well-formedness of dependency structures. The four axioms are:

1. Exactly one element is independent.
2. All other words depend directly on another word.
3. No word depends directly on multiple words.
4. If word X depends directly on Y and element Z intervenes between them (regarding the linear word order), then Z either depends directly on A, B or another intervening element.

The first two axioms cover the essential conditions for dependency tree structures (allowing exactly one root node). The third axiom additionally ensures single-headedness of all elements. The fourth axiom – the requirement of *projectivity* – forbids crossing edges in dependency trees and thereby deprives one major advantage of dependency grammars over phrase structure grammars.

To retain this advantage, Duchier (1999) proposes a non-projective DG. Well-formedness of dependency trees is defined without any reference to word order. This especially allows accurate analyses of languages with free word order.

4.3 HIDDEN MARKOV MODELS

Hidden Markov Models (see Baum & Petrie, 1966; Baum & Eagon, 1967; Baum & Sell, 1968; Baum, 1972) are used to model non-deterministic processes which pass unobservable – hidden – states. Such processes are called hidden Markov processes. All kinds of Markov processes obey certain rules, but the current state of a hidden Markov model is not observable (contrary to regular Markov models). Only the output of such a hidden process can be observed and used to derive knowledge about inner transitions. Hidden Markov Model (HMM)s are used to model various processes like pattern recognition (e. g. speech and gestures), part-of-speech tagging and protein classification in bioinformatics.

A Markov model consists of n states $\Omega = \{\omega_1, \ldots, \omega_n\}$ and a transition matrix A containing the transition probabilities between two states. Thereby,

$$a_{ij} = P\left(\omega_j\left(t+1\right)|\omega_i\left(t\right)\right) \tag{4.13}$$

denotes the probability that the system transitions from state ω_i into state ω_j at a discrete point in time t. The transition probabilities must satisfy the normalization constraint $\forall i : \sum_{j=1}^{n} a_{ij} = 1$.

All Markov models satisfy the *Markov property* (see Markov, 1954). This term denotes the property that the system's state $\omega\left(t+1\right)$ at point $t+1$ only depends on the directly preceding state $\omega\left(t\right)$ and thus, Markov processes are *memoryless*.

The state of HMMs cannot be observed directly, instead only the resulting effects can be studied. Whenever the system passes a state, an observable symbol $v \in K$ is emitted. Hence, after passing the complete model consisting of t states, a sequence $V^t = \{v_1, \ldots, v_t\}$ is generated.

Additional to the states Ω and the transitions A, emission probabilities B are defined for the probability $b_j(v_t)$ assuming that symbol v_t is emitted at point in time t and the system is in state $\omega_j(t)$:

$$b_j(v_t) = P\left(v_t|\omega_j(t)\right) \tag{4.14}$$

The emission probabilities are normalized regarding the sum of all emission probabilities for a given state: $\forall k \in K : \sum_k b_j(k) = 1$.

In order to apply HMMs to problems, two challenges have to be solved. The first problem is the proper estimation of the model's parameters: transition probabilities A and emission probabilities B. Baum et al. (1970) propose the so-called *Baum-Welch algorithm* which iteratively optimizes a model given a suboptimal model.

To employ a trained model to a problem raises the task: find the sequence of hidden states that generates the observed sequence V^t with the highest probability. This task can be solved using the Viterbi algorithm.

The Viterbi algorithm (see Viterbi, 1967; Forney Jr., 1973) finds the most likely path through a HMM. This *Viterbi path* contains the sequence of passed hidden states. Instead of testing all possible sequences for the one yielding the highest probability, a dynamic programming approach reduces complexity of this task significantly.

The key idea behind the Viterbi algorithm is to compute the most probable path of length t that generates V^t while the current machine state is i for a given model $\mathcal{M}$:

$$\delta_i(t) = \max_{\omega_1,\ldots,\omega_t \in \Omega^t} P(\omega_1,\ldots,\omega_t = i, v_1,\ldots,v_t|\mathcal{M}) \tag{4.15}$$

As the underlying model $\mathcal{M}$ is memoryless and the transition to the next state solely depends on the current state, the function $\delta_i(t)$ can be defined recursively:

$$\delta_i(t) = \max_{1 \leqslant j \leqslant N} \delta_j(t-1) \cdot a_{ij} \cdot b_j(v_t) \tag{4.16}$$

To obtain the complete sequence of passed states is of interest, the last state $\vartheta_i(t)$ needs to be memorized for each step. Starting with length $t = 1$ and

incrementing t until the complete observed sequence is processed, the best path of length t is computed. Afterwards, the sequence of passed hidden states can be reconstructed starting with the last state and sequentially going backwards using $\vartheta_i(t)$ to the beginning.

The Viterbi algorithm uses the complete context and thus, generates the globally best path of hidden states.

HMMs proved to achieve high accuracy scores for the task of part-of-speech tagging (see Sec. 7). Regarding this task, the Viterbi path provides the best part-of-speech tag sequence for an observed word sequence.

4.4 CLUSTERING ANALYSIS

Clustering analysis is an important method for machine learning problems. Clustering emerged from the field of explorative data mining and became a major technique in pattern recognition, information retrieval and NLP.

Clustering belongs to the group of unsupervised learning methods bearing great potential advantages compared to supervised methods. Especially in the field of NLP, efforts and costs raised by human annotation of training data for supervised machine learning approaches shows high potential for optimization. Algorithms for POS tagging (see Clark, 2003; Goldwater & Griffiths, 2007), word sense disambiguation (see Shin & Choi, 2004), semantic role labeling (see Baldewein et al., 2004) and grammar induction (see Klein, 2005) based on clustering were recently presented and achieve competitive results.

Clustering is the process of detecting groups within a given data set. Each object is assigned to a class (or cluster) so that objects of a cluster are more *similar* to each other than to objects outside the cluster. Contrarily to classification tasks, the number and properties of classes are a priori unknown. Similarity of objects is calculated with respect to certain properties of the objects. The properties of an object are presented as a so-called feature vector as similarity computation on vectors is much more convenient than on arbitrary objects. Instead of defin-

ing the similarity of objects, an inverse distance function $d(a, b)$ is defined to measure the distance between two objects a and b. An overview about the most commonly used metrics is given in Sec. 4.4.1.

Clustering methods can basically be divided into two classes: hierarchical clustering (see Sec. 4.4.2) and partitional clustering (see Sec. 4.4.3). Hierarchical approaches induce a hierarchical structure which is extracted in an iterative manner. Partitional approaches on the other side seek the best flat partitioning of the data.

Most clustering techniques belong to the group of *strict* clustering methods – they assign exactly one class to each object. Some object distributions cannot be separated that strictly without obtaining arguable class assignments. Thus, some approaches allow *outlier* object which are not assigned to a class at all (see Brito et al., 1997) or allow *fuzzy* assignment of classes (multiple classes) to one object (see Nock & Nielsen, 2006).

4.4.1 Metrics

As objects are described by feature vectors, distance measures are defined in the feature space $\mathcal{R}^n$. Cluster algorithms use distance measures to detect objects with the smallest distance in between. Similarity measures can also be applied as they can be transformed into a distance measure.

In the following, some well-established metrics are presented.

EUCLIDEAN DISTANCE The Euclidean distance measures the distance of the two points a and b in the vector space:

$$d(a, b) = \sqrt{\sum_i (a_i - b_i)^2} \tag{4.17}$$

SQUARED EUCLIDEAN DISTANCE Analogously, the squared Euclidean distance measures the squared distance of the two points a and b in the vector space:

$$d(a, b) = \sum_i (a_i - b_i)^2 \tag{4.18}$$

MANHATTAN DISTANCE The Manhattan Distance measures the sum of the absolute differences of all components of the feature vectors a and b:

$$d(a,b) = \sum_i |a_i - b_i| \tag{4.19}$$

MAXIMUM DISTANCE The Maximum Distance denotes the maximum distance regarding one component of the feature vectors a and b. Components with smaller differences are not taken into account:

$$d(a,b) = \max_i |a_i - b_i| \tag{4.20}$$

COSINE SIMILARITY Cosine Similarity measures the similarity between two vectors (see Sec. 4.2.2.5). It can be easily transformed into a distance measure (e. g. $d(a,b) = 1 - s(a,b)$) as it is normalized between 0 and 1.

$$s(a,b) = \frac{a \cdot b}{|a||b|} \tag{4.21}$$

4.4.2 Hierarchical Clustering

The emergence of clusters out of a set of objects can be seen from two different perspectives. It can either be defined as the task of iteratively merging the most similar objects[2] into a cluster or as separation task – dividing the set into smaller subsets. The first strategy is called *agglomerative* clustering. Each object is initially put into a single cluster and with each iteration, most similar clusters are merged until exactly one cluster remains that contains all objects. The opposite approach – *divisive* clustering – starts with all objects in one cluster. Recursively, each cluster is split until no cluster consisting of more than object exists. In either case, a hierarchy of the involved objects is induced.

The decision which clusters to merge or where to split requires a *linkage criterion*. A linkage criterion determines how to estimate the distance of two clusters based on the distances of the contained objects of each cluster.

2 In terms of clustering analysis, the expression *most similar objects* is used synonymously for *objects showing the smallest distance*.

COMPLETE LINKAGE Complete linkage defines the distance of two clusters A and B as the maximum distance between two contained elements.

$$\text{link}(A, B) = \max\{d(a, b) : a \in A, b \in B\} \tag{4.22}$$

SINGLE LINKAGE Contrary to complete linkage, single linkage defines the distance of two clusters by the smallest distance between two participating elements.

$$\text{link}(A, B) = \min\{d(a, b) : a \in A, b \in B\} \tag{4.23}$$

AVERAGE LINKAGE Average linkage is a tradeoff between the both criteria above. The distance is defined as the arithmetic mean of all object pairs $(a, b) \in A \times B$:

$$\text{link}(A, B) = \frac{1}{|A|\,|B|} \sum_{a \in A} \sum_{b \in B} d(a, b) \tag{4.24}$$

CENTROID LINKAGE Similar to average linkage, centroid linkage takes all objects of both clusters into account. Instead of averaging over all possible entity pairs, the centroids $\overline{a}$ and $\overline{b}$ are defined for both clusters A and B respectively. The distance is then defined as the distance between centroids:

$$\text{link}(A, B) = d\left(\overline{a}, \overline{b}\right) \tag{4.25}$$

These linkage criteria represent only some of the most prominent ones, several more exist (e. g. Minimum energy clustering, sum of intra-cluster variance, Ward's criterion or V-linkage). The choice of a proper linkage criteria influences results of clustering tasks decisively and thus, an appropriate criterion has to be picked according to the task's objective and specific properties of the data.

4.4.3 Partitional Clustering

Partitional Clustering aims to identify the best flat separation of the data regarding to certain measures. A representative of this kind of clustering algorithms is

k-means (see Lloyd, 1957; Steinhaus, 1957; MacQueen, 1967). The objective is to separate n objects into $k \leqslant n$ clusters $C = \{C_1, \ldots, C_k\}$. Thereby for each cluster, the sum of squared distances between each contained object and the centroid of the cluster should be minimized.

In a formal way: $\underset{S}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x_j \in S_i} |x_j - \mu_i|^2$.

Before iterating over two steps and successively improving the clustering, a set of k centroids (or *means*) have to be initialized. The centroids of the clusters $C_1, \ldots, C_k$ are denoted by $c_1, \ldots, c_k$ respectively.

After this initialization step, the following two steps are repeated until the clustering converges or the maximum number of iterations is met:

CLUSTER ASSIGNMENT Each object x is assigned to the cluster y with the closest mean ($\underset{y}{\operatorname{argmin}}\, d(x, c_y)$ with $1 \leqslant y \leqslant k$). Each object is assigned to exactly one cluster, although multiple centroids may be equally distant.

CENTROID CALCULATION The centroid c_i is re-calculated for each cluster C_i:

$$c_i = \frac{1}{|C_i|} \sum_{x \in C_i} x \tag{4.26}$$

One drawback of k-means is the possibility for convergence to a local minimum which is heavily discussed (see Pollard, 1982; Bottou & Bengio, 1995).

4.4.4 Evaluation Measures

Evaluation of clustering algorithms is a challenging task. For NLP, comparison of two clusterings is very important to verify accurate results (e. g. as for POS induction).

Evaluation techniques for clustering are often divided into internal and external evaluation. Internal evaluation is performed directly on the clustered data and measures scores like density and well-separateness of the data. Results of NLP algorithms shall often be compared to existing gold standards which are created manually and thus, provide a reliable basis for evaluation. This type of evaluation is called external evaluation.

Most proposed evaluation metrics can be assigned to either mapping-based measures or entropy-based measures. As there is no *perfect* measure, results for multiple measures are given to weaken certain metric-specific (dis-)advantages of clustering approaches.

4.4.4.1 *Mapping-based Measures*

Mapping-based measures try to map the induced clustering to a gold clustering. Basically, two measures are established, both can be applied using the Kuhn-Munkres Algorithm (see Kuhn, 1955; Munkres, 1957):

1-TO-1 MAPPING It measures the best cluster assignment accuracy under the condition that at most one created cluster is assigned to any gold cluster.

M-TO-1 MAPPING It measures the fraction of correctly assigned clusters under the condition that each created class is mapped to the gold cluster with which it most frequently co-occurs.

Further (more complex) mappings try to circumvent the drawbacks of these two measures (see Larsen & Aone, 1999; van Dongen, 2000; Zeng et al., 2002).

4.4.4.2 *Entropy-based Measures*

A shift to measures motivated by information yield can be observed in latest research. Consequentially, more measures based on entropy are proposed.

A contingency matrix A is defined for efficient computation and convenient entropy definition. $K = \{k_1, \ldots, k_{|K|}\}$ denotes the gold classes to which all N points are assigned in the gold standard. Analogously, $C = \{c_1, \ldots, c_{|C|}\}$ denotes the induced clustering. The contingency matrix A is then defined as $A = \{a_{ij}\}$ with the dimensions $|K| \times |C|$ such that a_{ij} is the number of objects that are members of gold class k_i and are assigned to class c_j by the clustering algorithm.

The following entropies can be defined over the data:

$$H(C|K) = -\sum_{k=1}^{|K|} \sum_{c=1}^{|C|} \frac{a_{ck}}{N} \log \frac{a_{ck}}{\sum_{c=1}^{|C|} a_{ck}} \tag{4.27}$$

$$H(C) = -\sum_{c=1}^{|C|} \frac{\sum_{k=1}^{|K|} a_{ck}}{n} \log \frac{\sum_{k=1}^{|K|} a_{ck}}{n} \tag{4.28}$$

$$H(K|C) = -\sum_{c=1}^{|C|} \sum_{k=1}^{|K|} \frac{a_{ck}}{N} \log \frac{a_{ck}}{\sum_{k=1}^{|K|} a_{ck}} \tag{4.29}$$

$$H(K) = -\sum_{k=1}^{|K|} \frac{\sum_{c=1}^{|C|} a_{ck}}{n} \log \frac{\sum_{c=1}^{|C|} a_{ck}}{n} \tag{4.30}$$

VARIATION OF INFORMATION The measure Variation of Information is proposed by Meila (2007). Least homogeneous clusterings result in maximum values for $H(C|K)$ and $H(K|C)$. For homogeneous clusterings, both entropy values decrease to 0. Consequently, better clusterings achieve *lower* VI scores (see Equ. 4.31).

$$VI = H(C|K) + H(K|C) \tag{4.31}$$

VALIDITY MEASURE The Validity Measure (see Rosenberg & Hirschberg, 2007) measures to which extend the criteria of *homogeneity* and *completeness* are satisfied. Homogeneity measures if only data points of a single gold class are assigned to a single cluster (see Equ. 4.32). Completeness measures if all objects of a gold class are assigned to a single cluster (see Equ. 4.33).

$$h = \begin{cases} 1 & \text{if } H(C,K) = 0 \\ 1 - \frac{H(C|K)}{H(C)} & else \end{cases} \tag{4.32}$$

$$c = \begin{cases} 1 & \text{if } H(K,C) = 0 \\ 1 - \frac{H(K|C)}{H(K)} & else \end{cases} \tag{4.33}$$

Based on these two criteria, the V measure is defined as the harmonic mean of homogeneity and completeness:

$$V = \frac{2hc}{h+c} \tag{4.34}$$

Values of V lie in $[0 \dots 1]$ and higher values imply superior clusterings.

NORMALIZED VARIATION OF INFORMATION The measure Normalized Variation of Information is a normalized version of Variation of Information (see Reichart & Rappoport, 2009) and thus, can be employed to compare different clusterings across different data sets. As it is based on VI, lower scores denote better clusterings.

$$\text{NVI} = \begin{cases} \frac{H(C|K)+H(K|C)}{H(C)} & H(C) \neq 0 \\ H(K) & H(C) = 0 \end{cases} \tag{4.35}$$

4.5 SUPPORT VECTOR MACHINES

A SVM (see Vapnik, 1995; Cortes & Vapnik, 1995) is a supervised machine learning algorithm. It is a binary classifier operating in a multidimensional space. SVMs use a maximum margin hyperplane to find the optimal linear separation of two classes.

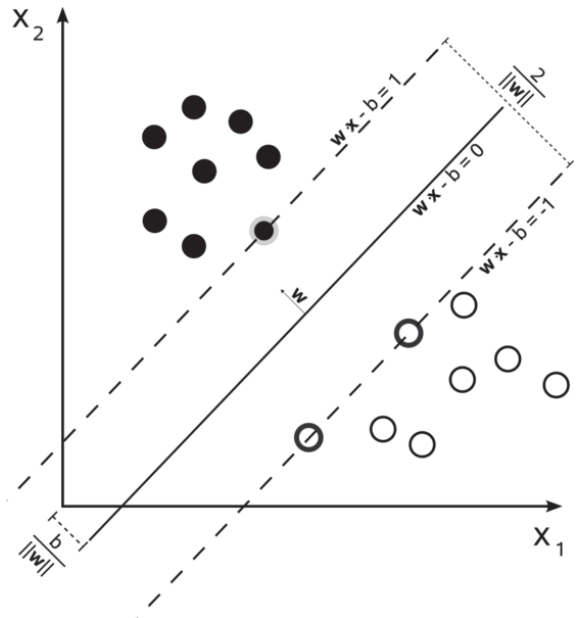


Figure 3.: SVM: Maximum separation hyperplane

Data points on the margin of the hyperplane are called *support vectors* (see Fig. 3 [3]).

Given the training instances T[4], a n-dimensional hyperplane $f(x)$ is defined as in Equ. 4.37 (see Press et al., 2007).

$$T = \{(x_i, y_i) : i = 1, \ldots, n\} \subset \mathbb{R}^d \times \{-1, 1\} \tag{4.36}$$

$$f(x) \equiv wx + b = 0 \tag{4.37}$$

As the hyperplane $f(x)$ should separate both classes, all training instances of one class (with $y_i = 1$) are on the opposite side as the instances of the other class (with $y_i = -1$). After determination of the normal vector w and the offset b, the hyperplane can be used as decision function to determine the class of new instances ($f(x) > 0$ and $f(x) < 0$, respectively).

To choose the hyperplane that separates the data in an optimal way, the maximum margin approach is applied. This means that the Euclidean distance between the support vectors and the hyperplane is maximized. This can be achieved by adjusting w and b so that:

$$\begin{aligned} wx_i + b &\geqslant +1 \qquad \text{if } y_i = +1 \\ wx_i + b &\leqslant -1 \qquad \text{if } y_i = -1 \end{aligned} \tag{4.38}$$

or reformulated

$$y_i(w \cdot x_i + b) \geqslant 1 \tag{4.39}$$

The distance d between the hyperplanes can be defined as (see Fig. 3):

$$d = \frac{2}{|w|} \tag{4.40}$$

3 http://en.wikipedia.org/wiki/Support_vector_machine

4 It is assumed that the training instances are linearly separable.

Instead of maximizing d with respect to $|w|$, $|w|^2$ can be minimized using the Lagrangian (see Schnitzler & Eitrich, 2006) and the condition 4.39:

$$\mathcal{L} = \frac{1}{2} w \cdot w + \sum_i \alpha_i (1 - y_i (w \cdot x_i + b)) \tag{4.41}$$

Equ. 4.41 can be reduced by calculation of the partial derivates for w and b and substitution. This leads to Equ. 4.42.

$$\mathcal{L}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \tag{4.42}$$

After estimation of α_i, w and b, the maximum margin hyperplane is obtained and the following decision function can be used to classify instances into one of the classes:

$$F(x) = sig \left(\sum_{x_i \in SV} \alpha_i y_i \, (x_i \cdot x) + b \right) \tag{4.43}$$

SVMs can be used to separate non-linearly separable data in two different ways. One possibility is the introduction of a *slack variable* ξ_i for each instance x_i. For every linearly separable data point, the corresponding slack variable ξ_i is zero, while it describes the amount of discrepancy for all other instances (see Press et al., 2007):

$$y_i (w \cdot x_i + b) \geqslant 1 - \xi_i \tag{4.44}$$

The calculation is analogous to linearly separable data and is skipped at this point.

Another possibility is the usage of *Kernel methods* (see Cristianini & Shawe-Taylor, 2000) which will be described in detail in the next section (see Sect. 4.6).

Multi-class classification can be achieved by application of multiple binary classifiers. An empirical study of some multi-class SVM configurations can be found in Duan & Keerthi (2005).

4.6 KERNEL METHODS

In order to separate non-linearly separable data the so called *kernel trick* (see Aizerman et al., 1964) can be applied. A function Φ is applied to transform the input data from the input space $\mathbb{R}^d$ into a higher dimensional feature space $\mathcal{H}$, so that a linear separation can be performed in $\mathcal{H}$:

$$\Phi : \mathbb{R}^d \to \mathcal{H} \text{ and } x \mapsto \Phi(x) \tag{4.45}$$

The hyperplane will be calculated in $\mathcal{H}$ due to easier linear separation. Scalar products $\langle \Phi(p), \Phi(q) \rangle$ for two instances p and q are necessary for the hyperplane calculation, but the complexity increases with the dimensionality of $\mathcal{H}$.

To avoid such difficult scalar product calculations, a kernel function can be defined in $\mathbb{R}^d$:

$$\mathcal{K}(p, q) = \langle \Phi(p), \Phi(q) \rangle \tag{4.46}$$

The calculation of scalar products in the high-dimensional feature space $\mathcal{H}$ is obsolete due to kernel functions. To ensure, that a function behaves as expected and can be employed as a kernel function, the criteria of *Mercer's theorem* (see Mercer, 1909) have to be fulfilled. A kernel function must be symmetric: $\mathcal{K}(p, q) = \mathcal{K}(q, p)$. Additionally, the kernel matrix K with $K_{ij} = \mathcal{K}(p_i, p_j)$ has to be positive and (semi-)definite for all training instances p_i:

$$a^t K a = \sum_{i,j} a_i a_j K_{ij} \geqslant 0 \quad \forall a \in \mathbb{R}^d \tag{4.47}$$

With the possibility of kernel methods, SVMs can be applied to more complex input data as just simple vector data. Hence, various problems such as classification of strings (see Lodhi et al., 2002), images (see Barla et al., 2002) or phylogenetic profiles (see Vert, 2002) can be approached.

Kernel methods show a unique property distinguishing them from other machine learning approaches: no features have to be generated or extracted out of instances. Instead, the original representation (e. g. parses with additional

annotations) of the instances is retained and similarity or kernel functions are applied directly to the data. This enables the machine learning approach to explore a much larger feature space than on generated feature vectors.

4.6.1 Tree Kernels

Syntactic features like parse trees can improve the performance of relation extraction approaches (see Miller et al., 2000). Defining kernel functions on syntactic trees enables classifiers to train and classify relation instances using comprehensive syntactic features like words, part-of-speech tags, syntactic tree structures and semantic roles (see Gildea & Jurafsky, 2002).

Many kernels have been proposed for application to relation extraction tasks, but despite the fact that all of them operate on syntactic tree structures, remarkable differences in feature selection exist.

A major distinction can per achieved by estimating the syntactic structure search space. Regarding the example sentence "Mary brought a cat" (see Moschitti, 2006), several syntactic trees can be created. The corresponding constituency parse tree is given in Fig. 4a, the dependency parse tree in Fig. 4b. Tree kernels for RE have been proposed for both constituency (see Collins & Duffy, 2001; Zelenko et al., 2003) and dependency parse trees (see Culotta & Sorensen, 2004; Bunescu & Mooney, 2005). Further considerations will focus on constituency trees as they yield superior results than dependency parse trees for the task of relation extraction (see Jiang & Zhai, 2007).

The idea behind tree kernel function is to detect and compare fragments of trees in order to calculate similarity. Three important characterization of tree fragments can be distinguished (see Moschitti, 2006):

SUBTREES Subtrees are obtained by selection of any node out of a tree with all its descendants. Fig. 5 shows all subtrees of the parse tree given in Fig. 4a.

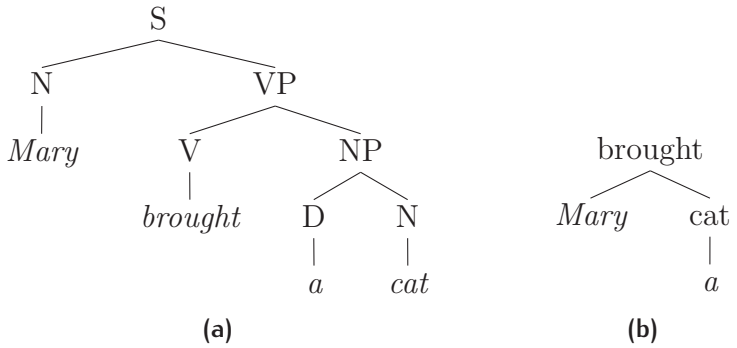


Figure 4.: Parse trees for "Mary brought a cat"

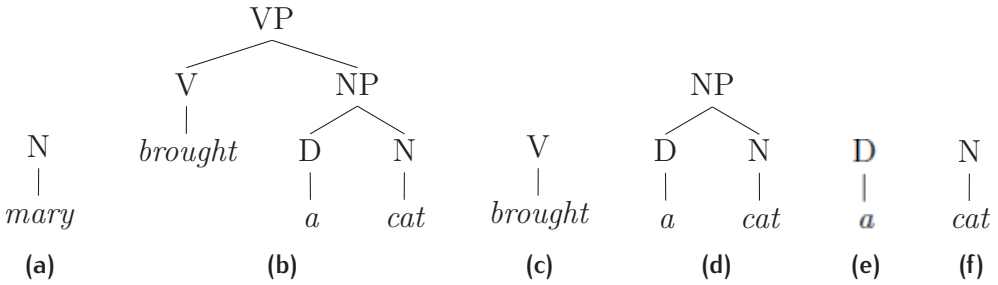


Figure 5.: All subtrees for "Mary brought a cat"

SUBSET TREES While grammatical rules will be satisfied, terminal symbols can be left out. Fig. 6 shows some subset trees of the VP given in Fig. 4a. Tree 6d matches all constraints of the rule NP ← D N, while discarding of D or N would violate the grammaticality of this tree fragment.

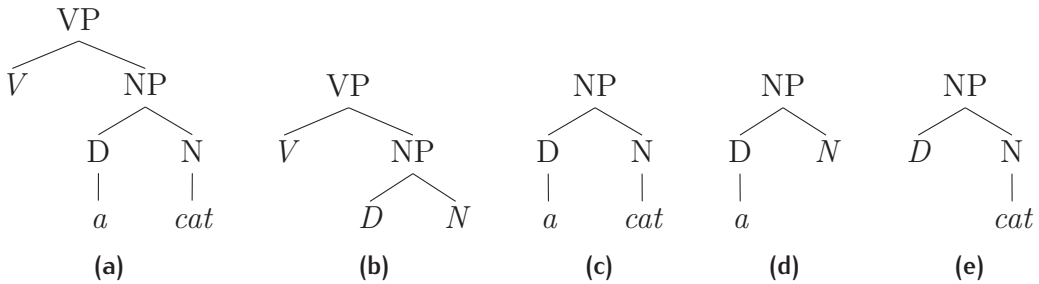


Figure 6.: Some subset trees for "Mary brought a cat"

PARTIAL TREES The most general form of tree substructures are partial trees. No constraints about grammaticality or completeness have to be matched leading to a very rich tree space (see Fig. 7a for violation of VP ← V NP). Some partial trees of the VP given in Fig. 4a are shown in Fig. 7:

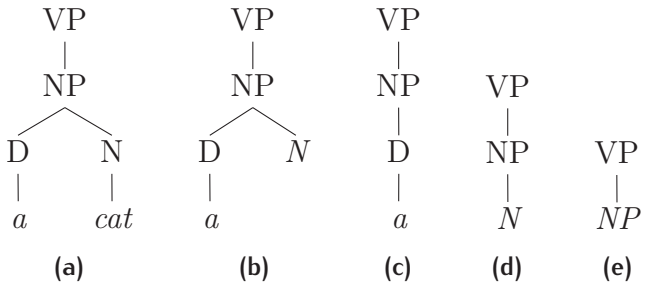


Figure 7.: Some partial trees for "Mary brought a cat"

The growing number of possible substructures represents the different information levels of each of the three tree spaces. In the following, four different tree kernels for RE will be introduced along with their feature selection strategies.

All presented kernels belong to the group of *Convolution Kernels* (see Haussler, 1999; Watkins, 1999). Convolution kernels are recursively defined kernels over fragments of a discrete structure. In the case of trees and the corresponding substructures, it is very convenient to calculate the score of a node recursively as a function of its descendants.

4.6.1.1 *Subset Tree Kernel*

The Subset Tree Kernel (STK) presented in Collins & Duffy (2001) is one of the first kernel functions applied to NLP problems. It uses subset trees to build a feature space:

$$\mathcal{F} = \{f_1, f_2, \ldots, f_{|\mathcal{F}|}\} \tag{4.48}$$

As all tree kernels aim to count common tree fragments in order to calculate similarity of two trees T_1 and T_2, an indicator function is proposed by Collins & Duffy (2001) to determine whether a tree fragment f_i of $\mathcal{F}$ is rooted under a certain node n:

$$I_i(n) = \begin{cases} 1 & f_i \text{ is rooted in } n \\ 0 & \text{else} \end{cases} \tag{4.49}$$

The inner product of two trees is defined by

$$K(T_1, T_2) = h(T_1) \cdot h(T_2) \tag{4.50}$$

with N being the node set of tree T and $h(T)$ denoting the count of contained tree fragments from the feature space:

$$h(T) = \sum_{n \in N} I_i(n) \tag{4.51}$$

The value of $K(T_1, T_2)$ depends strongly on the tree sizes. A normalization step is essential to ensure proper comparability between kernel scores obtained for trees of significantly different sizes. Lodhi et al. (2002) propose a suitable normalization function that still fulfills all criteria of Mercer's Theorem:

$$K'(T_1, T_2) = \frac{K(T_1, T_2)}{\sqrt{K(T_1, T_1)\, K(T_2, T_2)}} \tag{4.52}$$

This normalization is applied to all tree kernels.

4.6.1.2 *Contiguous Subtree Kernel*

The Contiguous Subtree Kernel (CSTK) functions presented in Zelenko et al. (2003) show different characteristics when being compared to other tree kernels. They do not belong to the class of convolution kernels *per se*, but are much related to these kernels. A more important difference is there inclusion of the requirements of relation extraction tasks. Their definition of nodes allows an arbitrary set of properties containing entity types and roles which are ignored by most kernel functions.

Zelenko et al. (2003) define two functions: a matching function t (see Equ. 4.53) estimating the matchability of two nodes and a similarity function k (see Equ. 4.54).

$$t(T_1.p, T_2.p) = \begin{cases} 1 & \text{if } T_1.\mathrm{Type} = T_2.\mathrm{Type} \text{ and } T_1.\mathrm{Role} = T_2.\mathrm{Role} \\ 0 & \text{otherwise} \end{cases} \tag{4.53}$$

$$k(T_1.p, T_2.p) = \begin{cases} 1 & \text{if } T_1.\mathrm{Text} = T_2.\mathrm{Text} \\ 0 & \text{otherwise} \end{cases} \tag{4.54}$$

Based on these two functions, the following kernel function is defined:

$$K(T_1, T_2) = \begin{cases} 0 & t(T_1.p, T_2.p) = 0 \\ k(T_1.p, T_2.p) + K_c(T_1.c, T_2.c) & otherwise \end{cases} \tag{4.55}$$

$K_c(T_1.c, T_2.c)$ is the similarity function on the child nodes of the trees T_1 and T_2, respectively. The definition of K_c decides which features space $\mathcal{F}$ is employed throughout the kernel score calculation.

For arbitrary tree fragments, K_c can be noted as

$$K_c(T_1.c, T_2.c) = \sum_{i,j,l(i)=l(j)} \lambda^{d(i)}\lambda^{d(j)} K(T_1[i], T_2[j]) \prod_{s=1,\ldots,l(i)} t(T_1[i_s].p, T_2[j_s].p) \tag{4.56}$$

where i and j are sequences of indices with $i_1 \leqslant i_2 \leqslant \ldots \leqslant i_n$ (j is defined analogously). The function $l(i) = |i|$ denotes the length of the sequence i, while $d(i) = i_n - i_1 + 1$ stands for the maximum distance of two elements of i.

In case of contiguous subtree kernel computation, only contiguous child subsequences need to be enumerated by K_c.

Formally, K_c[5] reduces to

$$K_c(T_1.c, T_2.c) = \sum_{i,j,l(i)l(j)} \lambda^{l(i)} K(T_1[i], T_2[j]) \prod_{s=1,\ldots,l(i)} t(T_1[i_s].p, T_2[j_s].p) \tag{4.57}$$

4.6.1.3 *Sparse Subtree Kernel*

Analogously to the contiguous subtree kernel presented in the last section, Zelenko et al. (2003) proposes the Sparse Subtree Kernel (SSTK). In this case, discontiguous child sequences are allowed with $d(i) > l(i)$.

4.6.1.4 *Partial Tree Kernel*

The Partial Tree Kernel (PTK) proposed in (Moschitti, 2006) operates in the partialtree space and thus, uses the richest search space of the four kernels regarding

5 A small change to notion: λ stands for λ^2 (as in Equ. 4.56)

syntactic features. It computes the number of common partialtrees in both trees T_1 and T_2. A kernel space $\mathcal{F}$ is defined containing all partialtrees of those trees.

Within space $\mathcal{F}$ an indicator function $I_i(n)$ is defined as given in Equ. 4.49. The kernel function $K(T_1, T_2)$ is defined as

$$K(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2) \tag{4.58}$$

where N_{T_1} and N_{T_2} are the node sets contained in T_1 and T2, respectively. The number of common tree fragments of two nodes n_1 and n_2 is denoted by:

$$\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} I_i(n_1) I_i(n_2) \tag{4.59}$$

The properties of all presented kernel methods will be analyzed in the next section. The focus of this discussion is the applicability to relation extraction tasks on automotive domain data.

5

CORPORA AND PREPROCESSING

In the following sections, the corpora being used for training and evaluation throughout this thesis are introduced focusing on their respective characteristics.

A major argument for unsupervised natural language processing is the existence of heterogeneous language resources of different language or type. Not only do they differ in language and dialect, textual resources from multiple domains and/or stylistic origin show distinct characteristics. Despite this fact, most NLP approaches are often trained and evaluated on high quality newspaper corpora which have been – in most cases manually – augmented with linguistic annotations (e. g. part-of-speech tags, parse trees, semantic roles, named entities and relations).

Obviously, this course of action seems to be inappropriate and inconsistent for experiments employing unsupervised methods. Thus, using several different data sources is predestinated for unsupervised approaches. This is probably the most meaningful evaluation to prove algorithms to be able to perform well on diversified textual resources.

This section deals with characterization of textual resources and appropriate measures. Most differences match one of the following categories:

LANGUAGE Most of the existing natural languages differ by nature. Important aspects with regard to syntactic analyses are word order, branching (right vs. left), inflection and compounding. For many languages several dialects exist showing significant differences in some cases even within a language.

ORTHOGRAPHY Correct spelling is available in most textual resources. But with the rising importance of the Web2.0, nearly unlimited amounts of textual data without such a high standard in terms of proper spelling and correct usage of case sensitivity is available. User generated data (e. g. forum posts, blog entries or chat logs) is often generated within very short time and without any need for quality assurance. Abbreviations are used very freely and in a very creative way during live communication where fast responses are considered to be more important than correct orthography. Even missing or additional whitespaces occur.

TERMINOLOGY High quality texts often contain more sophisticated terms and constructions than standard text. Depending on the domain, technical terms or common terms with a domain-dependent meaning are used. In the Web2.0, new words are invented very rapidly. A completely new *Internet slang* has been invented covering abbreviations, emoticons and the rapid distribution of *Leetspeak*.

STYLISTICS Some authors prefer certain constructions and expressions or tend to use longer sentences or short ones. These and similar features are used in fields like linguistic fingerprinting and author identification (see Holmes, 1994). The texts stylistics do also have huge impact on readability (see Gunning, 1952). This especially holds for domain language and data to avoid misunderstanding of facts or relations.

TEXT TYPE The structure of a document is sometimes predetermined by its text type. Letters have a fix structure and it is no surprise that they show more addresses than other types of text. In current NLP research, Wikipedia[1] articles are used as knowledge base (see Kazama & Torisawa, 2007; Sumida & Torisawa, 2008). Due to its fixed structure, it is convenient to crawl those articles and split them into parts containing the required information.

1 http://www.wikipedia.org/

5.1 MEASURES FOR LANGUAGE STATISTICS

Statistical measures can be used to describe language statistics in a quantitative manner. The objective is to compare the available corpora and to derive appropriate processing steps. Besides some general language statistics measures some special domain language measures are presented and considered. A comprehensive description of the following measures can be found in Schierle (2011).

In the following, a brief description of all necessary measures for corpus comparison is given:

ENTROPY The *Shannon entropy* denotes the information gain obtained by random experiment (see Shannon, 1948; Manning & Schütze, 1999). In language statistics, it presents a measure of information loss when a certain token is missing.

The given formula is normalized to the vocabulary size $|V|$:

$$H = -\sum_{t_i \in V} p(t_i) \log_{|V|} p(t_i) \tag{5.1}$$

INFORMATION DENSITY For information extraction purposes, a very meaningful measure is the *information density* of the corpus. With the scope automotive domain, relevant entities are amongst others components, symptom descriptions or corrective actions. The information density ID is defined as the ratio of the number of relevant tokens N_r and the corpus size N:

$$ID_{corpus} = \frac{N_r}{N} \tag{5.2}$$

MISSPELLING RATIO As most algorithms are trained on high quality newspaper corpora, misspellings cannot be observed (or very rarely). Regarding domain specific corpora which do not contain data from reviewed or revised sources misspellings occur more frequently. Schierle (2011) proposes the *spelling accuracy measure* SA_{voc} which is defined as the ratio of correctly spelled tokens and all tokens:

$$SA_{voc} = \frac{N_{cor}}{N} \tag{5.3}$$

SENTENCE LENGTH The average *sentence length* is a grammatical measure. The length $|S_i|$ of a sentence S_i is defined as the number of the tokens of S_i. The average sentence length $L(S_i)$ of a corpus is accordingly defined as:

$$L(S_i) = \frac{1}{|S|} \sum_{S_i \in S} |S_i| \tag{5.4}$$

GRAMMATICALITY A rather simple measure for *grammatical complexity* GC (see Schierle, 2011) calculates the ratio of function words N_f and non-function (meaningful) words N_m:

$$GC = \frac{N_f}{N_m} \tag{5.5}$$

PREDICTABILITY *Corpus predictability* is a measure for the ability to predict successor words (see Tesitelová, 1992). It is based on the entropy of a first order Markov source:

$$H(\mathcal{S}) = -\sum_i p_i \sum_j p_i(j) \log p_i(j) \tag{5.6}$$

The corpus predictability CP is defined as:

$$CP = 1 - \frac{H(\mathcal{S})}{H_{max}(\mathcal{S})} \tag{5.7}$$

where $H_{max}(\mathcal{S})$ is the maximum possible entropy. A high value of CP stands for high predictability meaning a high probability of correct successor word prediction.

VOCABULARY SIZE *Vocabulary size* is defined as the number of different types observed in the corpus normalized to the size of the corpus N:

$$R_{voc} = \frac{|V|}{N} \tag{5.8}$$

DISPERSION Another vocabulary-based measure is the *dispersion*. It measures the relative portion of low frequency types $V_{low} = \{t | t \in V \wedge f(t) < 10\}$ among all types V (see Schierle, 2011):

$$D_{voc} = \frac{|V_{low}|}{|V|} \tag{5.9}$$

CONCENTRATION The *concentration of vocabulary* (see Tesitelová, 1992) is defined as the relative portion of tokens of high frequency types $V_{top} = \{t|t \in V \wedge r(t) < 10\}$ and the corpus size N:

$$C_{voc} = \frac{\sum_{v \in V_{top}} f(v)}{N} \tag{5.10}$$

5.2 AUTOMOTIVE CORPORA

Two different corpora consisting of domain language are considered and analyzed throughout this thesis. Both corpora focus on the automotive domain, but from a different perspective.

The QUIS Corpus contains repair order texts and consists of problem descriptions by the customer along with analysis results and applied corrections by the corresponding technician.

The AIM Corpus on the other hand consists of internet forum entries. These texts contain everything about automobiles, even loosely related topics and discussions.

5.2.1 Quality Information System Corpus

The QUIS Corpus contains repair order texts. For each repair order a text is created consisting of basically three parts:

COMPLAINT The customer states a problem – commonly consisting of components and observed symptoms.

CAUSE The technician shortly describes the cause of the problem.

CORRECTION The technician describes the applied corrective actions.

The QUIS Corpus contains about 4.1 million English repair orders. A very restricted domain language is used containing a high portion of special tokens, such as mileage, codes and common numbers.

A distinct set of repair orders has been manually annotated and is used for evaluation. This set is referred to as QUIS Eval. It consists of 500 repair orders.

5.2.2 Automotive Internet Mining Corpus

With the possibility of discussions in the World Wide Web, many use this opportunity to exchange their problems, thoughts and concerns about automotive issues. In contrast to the QUIS Corpus, the field of discussion is much broader in this corpus. Besides similar problem descriptions, subjective opinions (e. g. "I don't like the new headlights.", soft quality issues (e. g. "Why can't I order these two features together?") and off-topic discussions (e. g. Formula 1) are covered by this corpus.

This requires further processing steps. Off-topic discussions should be detected and discarded as they do not convey valuable information for the automobile manufacturer. Additionally, sentiment analysis becomes essential to detect sentiment trends for discussed issues.

The AIM Corpus contains 8.6 million German sentences. A distinct set of 860 forum entries has been manually annotated to build the evaluation corpus AIM Eval.

5.3 NEWSPAPER CORPORA

Most approaches originating in the scientific field of NLP are trained and evaluated on newspaper corpora. This is, admittedly, reasonable as newspaper text is of very high quality. Linguistic theories can be easily examined and no or only little effort has to be put into data cleaning steps. To be comparable with other approaches, evaluations for some tasks (e. g. Part-of-speech tagging, syntactic parsing) are performed on identical corpora.

5.3.1 Penn Treebank

For English data sets, the Wall Street Journal Corpus (WSJ) of the Penn Treebank (see Santorini, 1990; Marcus et al., 1993) is a widely established data set. It contains written English newswire annotated with parts-of-speech and constituency parses.

For some evaluations, only parts of the WSJ are used. In order to obtain a better representation of the data for human learners, punctuation and empty elements are removed (see Klein, 2005). Empty elements are tagged by *-NONE-*, punctuation tags are *, . : “”-LRB- and -RRB-*.

As newspaper text is not similar to general spoken language, some efforts has been made in Roark (2001) to re-order words or to spell-out numbers to adapt the data to general language. These changes would improve induced grammars, but were left out due to comparability to other approaches (e. g. Klein & Manning, 2004; Klein, 2005; Bod, 2006a, 2007b).

Evaluations are performed on two versions of this corpus. *WSJ* contains all 49208 trees of the WSJ. A less complex sub corpus – referred to as WSJ_{10} – contains all sentences containing at most 10 tokens (after the removals described above). This smaller corpus contains 7422 sentences.

5.3.2 NEGRA Corpus

For evaluation on German language data sets, the NEGRA Corpus (NEGRA) (see Skut et al., 1998) is analogously prepared. NEGRA is annotated with part-of-speech tags from the Stuttgart-Tübingen Tagset (STTS) (see Schiller et al., 1995). Empty element tags start with an asterisk, punctuation tags are *$. $, $(and $)*.

Similar to the English data, evaluations are performed on either the complete NEGRA containing 20602 sentences or a reduced version – referred to as $NEGRA_{10}$ – consisting of 2175 sentences with length of at most 10 tokens.

5.4 CORPUS COMPARISON

Concluding the measureable differences of the three available corpus types, all presented measures have been calculated and collected in Figure 8[2]. The first corpus contains data from repair orders[3], the second corpus contains entries crawled from automotive internet fora[4] and the news corpus contains news texts from WikiNews[5].

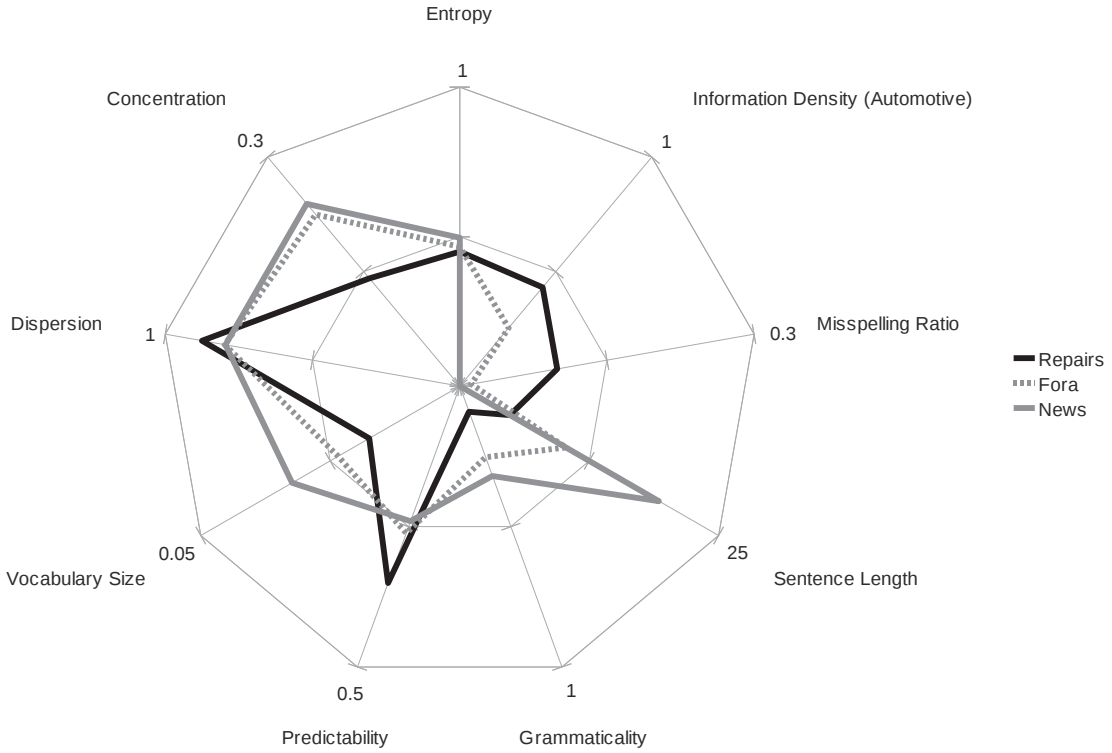


Figure 8.: Characteristics of different corpora

A comprehensive and thorough discussion can be found in Schierle (2011) and only the most important differences influencing information extraction are briefly discussed at this point.

2 see Schierle (2011)
3 see Section 5.2.1
4 see Section 5.2.2
5 http://en.wikinews.org

With increasing information density, syntactical complexity seems to decrease. While news texts show only a marginal amount of automotive information, grammaticality, sentence length and vocabulary size achieve the highest scores. Repair order texts on the other hand yield a very high portion of automotive information, only short sentences and restricted vocabulary and grammar complexity is used.

Regarding relation extraction based on syntactic features such as parse trees (e. g. kernel-based approaches), both corpora provide suboptimal conditions: repair order data cannot be parsed properly due to inaccurate POS tags and the very restricted language use, internet data contrarily shows very complex sentences containing many errors on various levels. Thus, approaches relying on accurate linguistic enhancements will perform worse than on high quality data (see Sec. 6).

Given these conditions, both corpora require extensive preprocessing to adjust the corpora's properties to newspaper texts as this kind of data can be processed more easily.

5.5 CORPUS PREPROCESSING

Before algorithms for RE can be applied, some preprocessing steps have to be run for both corpora. The tasks of the following preprocessing steps is to prepare both automotive corpora for Relation Extraction. This includes data cleaning steps along with linguistic analyses and semantic annotations.

Summarizing, these steps are:

TOKENIZATION Tokenization is the process of segmenting a text into meaningful tokens – in most cases into words (see Jackson & Moulinier, 2002). Domain corpora do contain a high portion of specific codes for certain information and thus, finding correct segments is more complex than it firstly appears to be. In the automotive domain, codes for repair actions, parts etc. are

very common along with mileage details (e. g. 20000km), prices for parts or information about consumptions. Especially all tokens consisting partly of numbers state a challenge for most tokenizer approaches. Thus, a rule based tokenizer method was chosen. Prioritized regular expressions are used to ensure high quality extractions (see Lehmann, 2006). Employing this approach, further processing steps benefit of additional information about the tokens, such as types (mileage, numbers, dates, part codes etc.) and semantic interpretations (e.g. 20k ml → 20000 miles).

SPELLING CORRECTION Due to the huge amount of spelling mistakes in repair orders, a spelling correction step is performed before further processing. Forum entries are no subject to spelling correction as they show only a small portion of mistakes (see Fig. 8). Besides typographical errors, a high percentage of (ambigouos) abbreviations exist. A context-sensitive approach is able to exploit the high predictability and to disambiguate abbreviations. The complete spelling correction is described in Schierle & Schulz (2007), it is based on Lehmann (2006).

ENTITY DETECTION In contrast to *proper nouns* recognized by most approaches to NER, automotive entities are not an open class. In fact, relevant entities are spread over multiple parts-of-speech. While *components* are nouns (e. g. *combustion engine*), *symptoms* can be nouns (e. g. *problem*), adjectives or adverbs (e. g. *inoperative*). Further categories like *locations* and *corrections* contain terms of different parts-of-speech, too. Those automotve entities can be enumerated and thus, it is not essential to be able to detect and extend the known entities extensively. An additional requirement is a hierarchical structure of all entities for structured reports and accumulation of results on different levels of abstraction. Using a wordlist-based approach can achieve all requirements if a hierarchical structure is realized as a thesaurus (see Schierle & Trabold, 2010; Schierle, 2011).

SENTIMENT ANALYSIS Recent research on *sentiment analysis* focuses on classification of words (see Hatzivassiloglou & Mckeown, 1997), sentences (see

Kim & Hovy, 2004, 2005) or even documents (see Pang et al., 2002). Both rule-based and machine learning methods are employed achieving reasonable results. But all of them state the assumption that the textual unit in question (be it a word, sentence or document) contains exactly one topic. This assumption is true on word level, but its substance may be doubted on sentence and document level (except for certain text types, e. g. product reviews). Within blog entries or forum discussions the corresponding author often addresses more than one topic (e. g. *I love X, but Y seems a little weird.*). The same observation can be made on technical reports describing customers' complaints which often cover more than one problem.

In order to provide a fine-grained analysis along with relations between opinion holders and opinion targets, sentiment analysis has to be performed on a deeper level – the phrase level. This is crucial for topic-centric sentiment analysis that is able to extract opinions and their respective targets (e. g. *X* and *love*, *Y* and *little weird*).

The approach of Remus & Hänig (2011) is employed as it achieves accurate results and is easily extendable due to its flexible rule based and yet language-independent architecture. The Polarity Composition Model (PCM) is based on findings of related fields of research (e. g. psychology) and its design is empirically validated by surveys with human subjects.

The Polarity Composition Model is designed as a two-layered polarity analysis – one level dealing with words, the other one with phrases relying on the first. The objective is to assign a word's prior polarity without any given context (see Wilson et al., 2009). A dictionary containing words, their inflected forms along with the prior polarity for each word is *SentiWS* (see Remus et al., 2010) which is used for word level sentiment analysis on the AIM Corpus. In contrast to other dictionaries, words are not classified into two classes (positive and negative, see Hatzivassiloglou & Mckeown, 1997), instead polarity values $v \in [-1, 1]$ express *intensity* classifications where a bigger $|v|$ identifies more *expressive* words.

Abbreviation	Category	Examples
ADJ	Adjectives, adverbs	superb
N	Nouns	displeasure
V	Verbs	despair
NEG	Negations	not
INC	Strengthening Intensifiers	very
DEC	Weakening Intensifiers	barely

Table 2.: Categories and their abbreviations

Still on word level, each word is put into a category to distinguish between *potential polar words* and *modifying words* (Polanyi & Zaenen (2006) propose the terminus Contextual Valence Shifters (CVS)). Assigned POS tags are used as light-weight *word sense disambiguation* (see Wilks & Stevenson, 1998). The category layer separates syntactic functions of the words from their semantic tasks. Table 2 gives an overview about these categories.

Phrase-level polarity analysis uses the results of word-level polarity analysis and thus, word level polarity analysis serves as abstraction from words and language. The objective of phrase level polarity analysis is to identify a word's *contextual polarity* (see Wilson et al., 2009). This is achieved by taking the local context into account – e. g. the (linguistic) *phrase structure*.

In order to compose the polarity of a phrase out of its contained lexical units, a bottom-up approach following the *compositional principle* is used (see Moilanen & Pulman, 2007; Neviarouskaya et al., 2009).

Each rule r has the form

$$r := [(d, f, p)\, CAT_i \ldots CAT_j] \tag{5.11}$$

where CAT_k is contained in {ADJ, N, V, NEG, INC, DEC} or CAT_k is a rule by itself. Additional operators like ... (an optional marker for disconti-

nuity), $\{CAT_k\}$ (a marker for optional categories), direction $d \in \{\rightarrow, \leftarrow\}$, an aggregation function $f \in \{a_+, a_*\}$ and a parameter $p \in \mathbb{Q}$ (obligatory for $a_\times$) build the formal environment to build complex formal constructs. A visualization of a rule is shown in Figure 9.

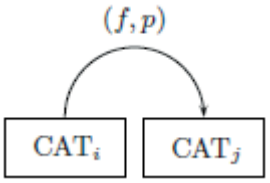


Figure 9.: Visualization of the formal construct

Because of the formal construct's *recursive definition*, the creation of rules of arbitrary complexity is possible. An example for polarity composition is given in Fig. 10:

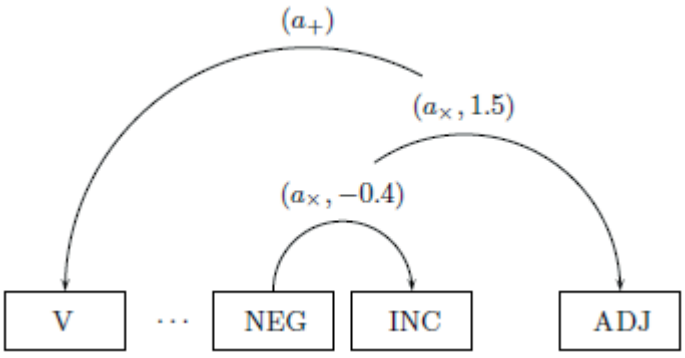


Figure 10.: Visualization of polarity composition

A comprehensive discussion about design, modeling decisions and empirical validation of the PCM is given in Remus & Hänig (2011).

6

ANALYSIS OF STATE-OF-THE-ART ALGORITHMS

With the growing number of annotated language resources supervised methods for NLP achieve more and more accurate results. Models for many languages exist and hence, supervised methods can be applied to many tasks. There is an argument about domain adaptation of supervised approaches. Many works focus on domain adaptation (for automatic taxonomy creation, see Cimiano et al., 2003b) to bridge the gap between supervised machine learning methods and application on real-world data and tasks. Remarkable results are achieved, but most of the approaches rely on high quality data which is not always available in certain environments (see Sec. 5.2).

In this section, state-of-the-art algorithms are analyzed to understand their strengths and weaknesses when being applied to the task of information extraction in the automotive domain and the respective textual data.

6.1 AUTOMOTIVE ENTITY TYPES

Information Extraction for the automotive domain focusses on an enumerable set of entities. Table 3 lists the most important ones which will serve as target entities throughout the evaluations of information related algorithms.

Those entities are detected as described in Sec. 5.5. It is very obvious that these entities do not belong to the well-studied set of named entities which are not in the scope of corporate quality analysis.

Entity type	description
COMPONENT	parts of vehicles, e. g. *wheel*, *gas pump*
SYMPTOM	problem descriptions, e. g. *problem*, *defect*, *inoperative*
CORRECTION	corrective actions, e. g. *changed*, *fixed*
CONDITION	certain conditions, e. g. *when raining*, *in the morning*
LOCATION	locations all over the vehicle, e. g. *front*, *driver side*
NEGATION	negations of all kind, e. g. *not*, *no*
VEHICLE	any mention of models, e. g. *A4*, *BMW*, *C Class*, *Golf IV*
SENTIMENT	polar phrases, e. g. *nice feature*, *not desirable*, *stunning model*

Table 3.: Target Entities for the Automotive Domain

6.2 AUTOMOTIVE RELATION TYPES

Arbitrary relations consist of a type and the corresponding entities (with their respective types). The entities embody certain roles of the relation and fill the assigned slots (e. g. the relation *is-president-of* accepts entities for both of the assignable roles *president* and *country*).

Automotive relation types show slightly different characteristics. Roles of entities (and the respective semantics) are directly defined by the category of the entity. Hence, the relation type is implicitly defined by the types of the related entities. A listing of all automotive relation types being evaluated on is given in Table 4.

Most of the relations provide insight into the data being useful for classic quality assurance analyses. The latter two relation types can be exploited by modern market research exploring the customer's opinion.

Relation type	description
COMPONENT – SYMPTOM	defect component and the observable symptom
COMPONENT – CORRECTION	corrective action applied to a defect component
COMPONENT – LOCATION	the location of a component within the vehicle
NEGATION – SYMPTOM	negation of a symptom
VEHICLE – SENTIMENT	polar utterance about a vehicle
COMPONENT – SENTIMENT	polar utterance about a component

Table 4.: Target Relations for the Automotive Domain

6.3 SUPERVISED POS TAGGING

Part-of-speech tagging is the assignment of parts-of-speech to a sentence's words (see Manning & Schütze, 1999). POS tags are essential for further processing like parsing and relation extraction. Highly sophisticated approaches (see Schmid, 1994; Brants, 2000; Collins, 2002) achieve more than 95% accuracy on token level for this task.

This performance seems close to perfection, but assuming an average sentence length of 20 still results in one incorrect tag per sentence.

In the following analysis on automotive data the *Stanford Tagger* (see Toutanova & Manning, 2000; Toutanova et al., 2003, accuracy of about 97%) serves as representative for supervised part-of-speech tagging approaches. Schierle (2011) evaluated the performance of five POS taggers on QUIS Corpus data and the Stanford Tagger achieved the most accurate results.

A more detailed introduction to the task of POS tagging can be found in Sec. 7.

6.4 SUPERVISED PARSING

Parsing is the task of assigning a *syntactical structure* to a sentence (Surdeanu et al., 2008). This structure is usually represented as a syntactic tree providing word order and hierarchical structure of phrases or dependencies (depending on the chosen grammar theory, see Section 4.2.4).

Most approaches to parsing use entropy maximization (see Charniak, 2000), maximum-likelihood estimation (see Klein & Manning, 2003a) or are based on Markov rules or HMMs (see Collins, 2003; Klein & Manning, 2003b). Due to the extensive research in the field of natural language parsing, a variety of models exist and are freely available for many languages[1].

For the following experiments, the Stanford Parser is used (see Klein & Manning, 2003a,b) as it yields accurate results and comes with available models for various languages.

6.5 TRAINING OF RELATION EXTRACTION KERNELS

In order to train a classifier for each relation type of interest, the training sentences are preprocessed as described in Section 5.5. This includes proper tokenization, spelling correction (for QUIS Corpus data only), entity detection and sentiment analysis (for AIM Corpus data only). Additionally, all instances are augmented with POS tags and constituency parses as described in the two preceding sections.

Starting with an entry out of the QUIS Corpus (see Fig. 11a), preprocessing expands abbreviations context-sensitively (e. g. *C/S* → *customer states* and *LT* → *left*[2]) and corrects typos and misspellings (e. g. *DRIVNIG* → *driving*). All characters are transformed into their respective lower-case versions in order to

1 e. g. for the Stanford Parser, see http://nlp.stanford.edu/software/lex-parser.shtml

2 *LT* is a common abbreviation for *left* and *light*. Context-sensitive disambiguation is able to pick the correct expansion.

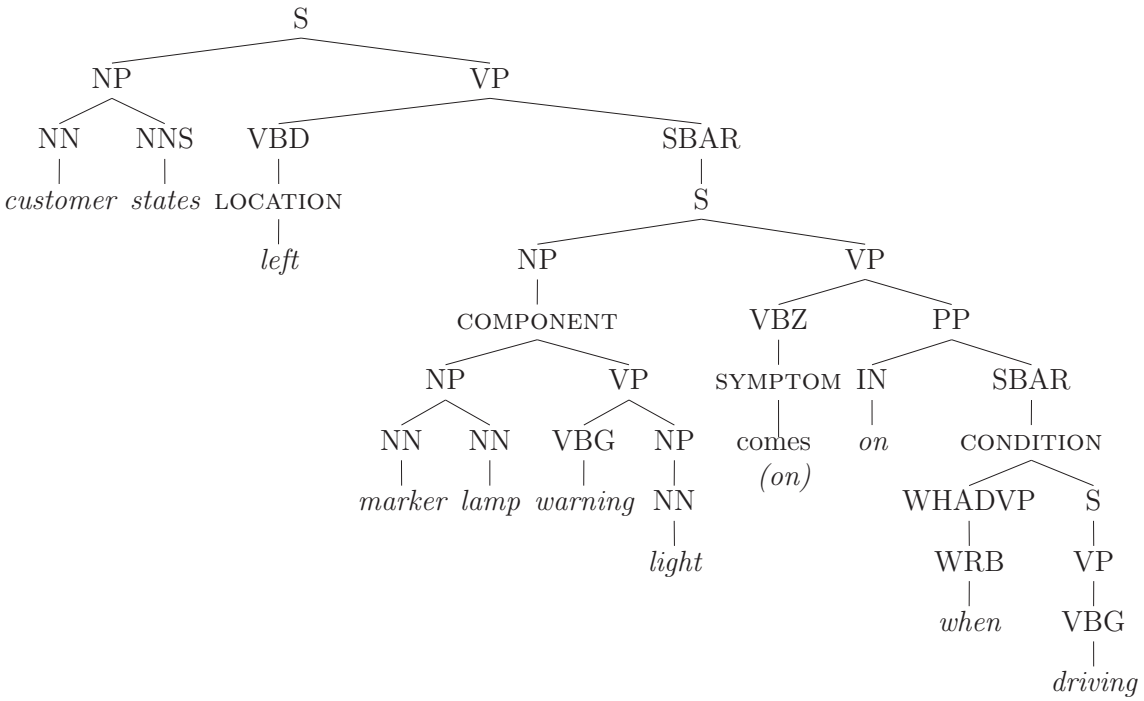
obtain better tagging results. Leaving the words upper-case results in tagging nearly every word as proper noun or normal noun. The cleaned entry of this example is given in Fig. 11b:

C/S LT MARKER LAMP WARNING LIGHT COMES ON WHEN DRIVNIG

(a) original entry

customer states left marker lamp warning light comes on when driving

(b) preprocessed entry



(c) parsed and annotated entry

Figure 11.: QUIS Corpus example

Although this example seems very easy to process as humans understand it without any trouble, state-of-the-art algorithms make mistakes. Incorrectly assigned POS tags (e. g. *NNS* ← *states* and *VBD* ← *left*) entail parsing errors (e. g. *left* is an adjective of COMPONENT *marker lamp warning light*). The completely processed entry is shown in Fig. 11c. The entity annotations (e. g. LOCATION, COMPONENT, SYMPTOM and CONDITION) are integrated into the tree for improved readability and do not alter the tree coming from the parser in the real application.

For each relation type, training instances were generated in a semi-supervised way. An automated candidate selection step extracted all sentences containing exactly two entities of the necessary types regarding the target relation. From this candidate set 200 instances were selected to form the positive training instances of a relation. Additionally, 200 negative instances were randomly selected. One half of them contain another relation type, the other half contains no relation at all and less than two entities. A binary SVM is trained for each relation type using the set of positive and negative instances. The implementation of *libSVM* (see Chang & Lin, 2011) has been used for training and classification. The results of the experiments on data from QUIS Corpus and AIM Corpus are given in the following section.

6.6 EVALUATION RESULTS AND DISCUSSION

In this section the results of the relation extraction evaluation employing kernel methods are presented. All methods are tested on two data sets: QUIS Eval and AIM Eval, the results are given in Table 5 and Table 6, respectively[3]:

Relation	Kernel	Precision	Recall	F-Score
COMPONENT – CORRECTION	STK	0.0476	0.6273	0.0884
COMPONENT – SYMPTOM	STK	0.0245	0.4828	0.0466
COMPONENT – LOCATION	STK	0.3333	0.0123	0.0238
NEGATION – SYMPTOM / CONDITION	STK	0.0007	0.0200	0.0014

Table 5.: Relation Extraction results on QUIS Eval

Although kernel methods proved to achieve very accurate results on newspaper test sets, this is not the case for the available data from the automotive domain. This originates in characteristics of the data, peculiarities of the relations

3 Only results for the kernel method achieving the best results are shown.

Relation	Kernel	Precision	Recall	F-Score
COMPONENT – CORRECTION	PTK	0.0502	0.1204	0.0708
COMPONENT – SYMPTOM	PTK	0.0667	0.0685	0.0676
COMPONENT – LOCATION	PTK	0.0385	0.4286	0.0707

Table 6.: Relation Extraction results on AIM Eval

and feature selection of the kernel methods. These reasons will be analyzed in the following.

6.6.1 Arbitrariness of Automotive Relations

Logical relations that are studied in most related work often connect named entities. Named entities are mostly expressed through nouns, noun phrases or other nominal expressions (except for numbers, temporal expressions, etc.). Entities of the automotive domain do not show this property. Most components occur as nominal expressions, but symptoms and corrections are mostly expressed as verbs and adjectives. Some concepts of the class symptom even have mixed nominal and verbal terms consisting of multiple tokens (e. g. *no problem found*).

As consequence automotive relations are not typically expressed as simpler relations like *is-a* (e. g. *X is a Y*, *X as other Ys*, etc.) or *is-CEO-of* (e. g. *X (head of Y)*, *X became CEO of Y*, etc.). Often, entities of a relation (e. g. COMPONENT – SYMPTOM) succeed each other directly. For these instances, only little contextual information is given. Typical verbs are not available (e. g. *makes* as in *engine makes some noise* is often discarded in repair orders and the utterance is reduced to *engine noisy*) making detection of such relations using predefined patterns difficult. Unsupervised pattern-based approaches as in Etzioni et al. (2005) automatically extract describing patterns for relations. They create characteristic patterns using the context before, between and after both entities through detec-

tion of descriptive words. As there are no words in between the involved entities, the most important information is absent.

RE seems so be trivial on directly succeeding entities, but repair orders are covered by 83% with automotive entities (see Hänig & Schierle, 2009) leading to many sequences of multiple directly succeeding entities. E. g. the example in Fig. 11 contains such a contiguous entity sequence (LOCATION COMPONENT SYMPTOM CONDITION). It is difficult to decide which entity is related to which other entity as no information except for the entity sequence itself is given. Thus, is it nearly impossible for pattern-approaches to solve this problem and kernel methods need a enormous amount of training examples which is not available[4] for such data sources.

The opposite case exists, too. Entities occur in manifold expressions and high-distance relations. This can be observed in entries from internet fora (see AIM Corpus, Section 5.2.2).

Three examples demonstrate the complexity (Fig. 12):

Is it possible the COMPONENT *transmission* is SYMPTOM *stuck in a higher gear* and that is what is causing the SYMPTOM *poor acceleration*?

(a)

The SYMPTOM *common problem* with first generation VEHICLE *W220s* is the COMPONENT *airmatic suspension*.

(b)

The COMPONENT *suspension* is NEGATION *not* SYMPTOM *malfunctioning*, but I hear a lot of SYMPTOM *road noise*.

(c)

Figure 12.: AIM Corpus examples

All three examples contain at least one relation of the type COMPONENT – SYMPTOM . While the first relation (COMPONENT *transmission* is SYMPTOM *stuck in a*

4 Creation of such a tremendous training set is to costly and time consuming for real world applications as it needs to be done whenever a new data source should be integrated or another language should be studied.

higher gear) is expressed using a basic pattern (COMPONENT is SYMPTOM), most of the others need more complex patterns to be matched. Many related entities occur within a distance of ten or more words creating a landscape of numerous patterns. Thus, pattern-based approaches can achieve high precision results, but recall will suffer as only few instances will be covered even with a high number of patterns. Feldman & Rosenfeld (2006) suggest patterns containing wildcards to improve recall. Their method detects identical words in different patterns to extract the denoting terms. As discussed above, denoting terms are very rare for automotive relations and the available denoting terms (e. g. *is*) are to generic and need a reliable verification.

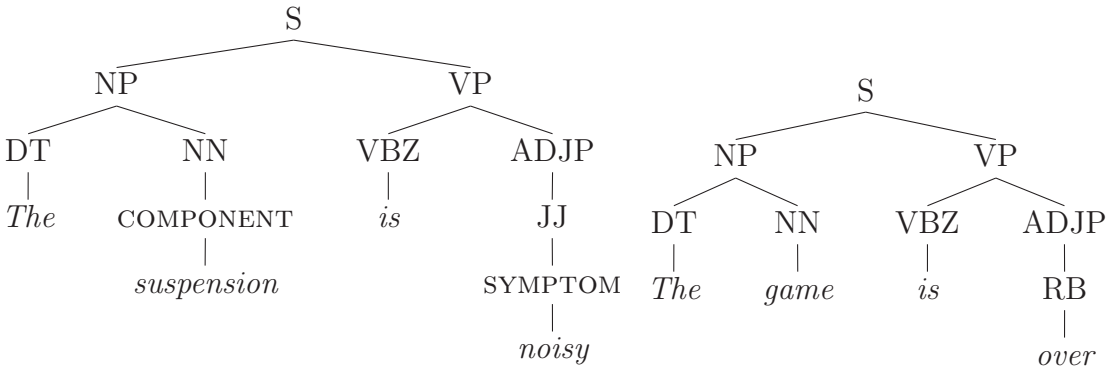
6.6.2 Feature Selection of Kernel Functions

Proposed kernel functions for RE mainly employ syntactic features to calculate similarities between two instances. The kernel's feature space influences the performance decisively. Zelenko et al. (2003) argues in favor for richer feature spaces (SSTK outperforms CSTK), but Moschitti et al. (2006) presents a deeper investigation. Experiments indicate that kernels operating on partial tree spaces perform better on dependency trees while kernels show superior results on constituency parses when using subset tree feature spaces.

This effect is grounded in the subtree space. Both subset trees and partial trees contain tree fragments without lexical items.

The example given in Fig. 13a contains an automotive relation (COMPONENT *suspension* – SYMPTOM *noisy*). A sentence with completely different semantic contents but still nearly identical syntactic structure (see Fig. 13b) shares numerous subset trees (see Fig. 14a) and partial trees (see Fig. 14b).

According to the kernel function definitions, which basically count common tree fragments, these two trees achieve a kernel score signifying similarity. This results in many overpropositions and thus, reduces precision. Especially forum entries contain long sentences which provoke a high portion of similar subtrees without sharing semantic information. Employing a contiguous subtree kernel

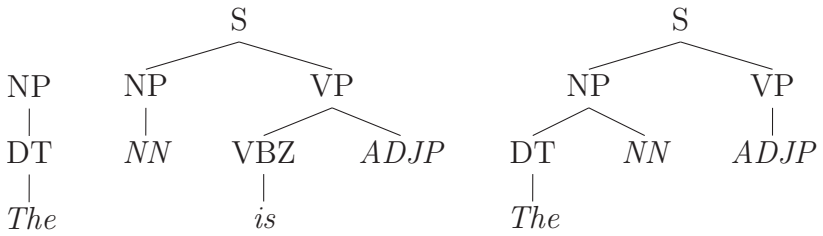


(a) Sentence containing an automotive relation

(b) Sentence without automotive information

Figure 13.: Similar trees with and without automotive relation

(a) Some shared subset trees



(b) Some shared partial trees

Figure 14.: Shared tree fragments

avoids this problem, but this feature space suffers from similar problems as patterns – low coverage on diverse expressions.

One possibility is to mind the involved entities' types in order to filter sentences without corresponding information. Zelenko et al. (2003) therefore proposes a *matching function* (see Section 4.6.1.2) to increase the importance of involved entities. This matching function forbids matching of nodes containing entities of either different roles or types. Although Zelenko et al. (2003) introduce a customized kernel for relation extraction that improves processing of relation instances, their matching function only weakens the problem. Syntactically

similar nodes without role or type annotations still are matched and increase similarity scores of analyzed instances.

6.7 CONCLUSIONS

Available algorithms for Relation Extraction are not applicable to the data contained in both automotive corpora. Existing part-of-speech taggers achieve reasonable results when trained on a reduced tagset for this data (see Schierle, 2011), but creation of training data for parsers is to costly and time consuming as parsers need much more annotated data to be trained properly. Syntactic augmentation yields valuable features for RE and thus, unsupervised methods for part-of-speech tagging and parsing should be studied to obtain syntactic enrichment of the data.

Existing kernel functions do not achieve good results when integrated into this RE workflow of the automotive domain. Typical tagsets (see Sec. 4.2.3) contain less than 100 tags. Unsupervised methods often induce significantly more word classes leading to fewer matches of syntactic similar tree fragments due to too fine-grained tagging.

In the following sections, appropriate algorithms for unsupervised part-of-speech tagging (see Section 7), parsing (see Section 8) and relation extraction (see Section 9) will be presented. While the first two algorithms deal with a general formulation of the respective NLP task, the latter deals with a sophisticated algorithm regarding the automotive domain.

Part II

UNSUPERVISED RELATION EXTRACTION

7

PART-OF-SPEECH TAGGING

Part-of-Speech tagging is the task of assigning parts-of-speech to words (see Manning & Schütze, 1999). Parts-of-speech are linguistic categories (see Sec. 4.2.3) on the word level and denote groups of words with similar syntactic properties and functions. Besides numerous supervised approaches to POS tagging (e. g. employing entropy maximization as in Toutanova et al. (2003)), unsupervised POS induction algorithms gain a lot attention in recent research. Despite the fact that results of unsupervised POS induction improve steadily, one major difference will remain due to the nature of unsupervised algorithms: the cluster labeling. Supervised approaches use manually annotated data for training and learn to correctly replicate this tagging on new unseen data. The trained tagger accordingly assigns exactly those tags of a specific tagset provided in the training data. Unsupervised algorithms do not rely on annotated data and do not have the possibility to learn which tags exist in a conventional way. The output of an unsupervised tagger is the assignment of word clusters to words. Words contained in a cluster implicitly define the part-of-speech although it is not denoted by *NN*, *DT* or other commonly used labels.

Most word types mainly bear one syntactic function and consequently, are tagged by one POS tag. Some word types accomplish different syntactic functions and are tagged by various tags depending on the current instance's function (e. g. *this* can be a *determiner* or a *pronoun* depending on the context). Nevertheless, there is a predominant tag word most effected word types (see Manning & Schütze, 1999) and this is why many approaches ignore proper disambigua-

tion. Experiments show comparable results achieved by simple approaches that always assign the most frequent tag of a word type (see Schierle, 2011).

In this section, several approaches to unsupervised POS tagging are presented. Afterwards, an extension for improvement of any unsupervised POS tagging algorithm is introduced. Different syntactic functions are disambiguated for certain word types depending on the tagging without proper disambiguation.

7.1 POS TAGGING – RELATED WORK

With the growing amount of internet texts, the need for unsupervised robust processing is growing analogously to avoid time-consuming annotation of training data when adapting to a new genre or domain. This general statement is also true for POS tagging, which can be seen as one of first NLP tasks being approached in a completely unsupervised manner.

All approaches to unsupervised POS tagging rely on context clustering in any form and thus, are based on the distributional description of word types as initially proposed by Schütze (1995). This idea is based on paradigmatic relations. Word types occurring within similar contexts (regarding local neighborhood contexts) show similar syntactic functions. This observation is exploited by all approaches, although different definitions of how exactly the appropriate context looks like are employed. Further disambiguation of existing algorithms can be achieved by looking at the applied cluster algorithms to eventually cluster word types into word classes.

Current approaches additionally incorporate more features, such as morphology (see Abend et al., 2010).

7.1.1 Distributional POS Tagging

Distributional POS Tagging (see Schütze, 1995) uses vector representations of a word types' context. Therefore, four different feature vectors are created to describe a word w:

- the left context of word w,
- the right context of word w,
- the right context of the word preceding w, and
- the left context of the word following w.

Each of the four context vectors consists of frequencies counting how often a context word cw occurs within the defined position relatively to w. The motivation of the first two vectors is obviously grounded in direct neighborship and thus, contributing significantly to the clarification of the word's syntactic function. The latter two represent properties of the word's context. They express the expectation of the context which syntactic category should appear at the position of w. In order to assign the correct tag to the word *work* in *her work seemed to be important* the fact that a noun phrase is expected in front of *seemed* is important, while the right context of *seemed* is not (see Schütze, 1995). Analogously, the right context of the left context word of w contributes to disambiguation of the proper word class.

As the dimension of each feature vector would be of the size of the lexicon, only the 250 most frequent words are taken into account. Hence, four feature vectors each containing 250 components are created.

In order to reduce data sparseness and influence of some idiosyncrasies (e. g. phonological constraints after *a* and *an*), a Singular Value Decomposition (SVD) is applied. All contexts vectors are concatenated and represented as matrix C. C consists of n rows where n is the number of different word types that will

be clustered and $k = 250$ columns, one for each possible context word. SVD is applied to decompose C as in:

$$C = T_0 S_0 D_0' \tag{7.1}$$

The diagonal matrix S_0 contains all singular values of C in descending order. All matrices are restricted to their first $m < k$ ($m = 50$) columns representing the principal components. The product of the resulting matrices T, S and D′ then represent the optimal feature matrix $\hat{C} = TSD'$ of rank m.

Finally, the cosine measure is applied to calculate similarities between two feature vectors during the clustering process. All accounted word types are classified into $c = 200$ word classes using Buckshot clustering (see Cutting et al., 1992).

7.1.2 SVD2

The approach presented in Lamar et al. (2010b,a) can be seen as an iterated version of Distributional POS Tagging.

Left and right context vectors are created using the most frequent 1000 word types. The additional context vectors (right context of the preceding context word and left context of the following context word) are not used for this approach. Analogously, SVD is applied to reduce the feature dimensions down to $m_1 = 100$. Afterwards, k-means clustering is applied to cluster all words of the corpus into $k_1 = 500$ classes (using cosine similarity).

A second pass of this process is carried out – using generalized contexts this time. Generalized contexts are calculated exactly like the other contexts described above, but instead of counting context words, the corresponding classes induced during the first iteration are counted. This kind of context generalization has been previously proposed by Schütze (1995), but he found that "one cannot conclude with certainty that generalized context vectors induce tags of higher quality". During the second pass, the matrices are reduce to rank

$m_2 = 300$ and k-means is applied to cluster the word types into k_2 clusters, where k_2 denotes the desired number of tags.

This iterated version outperforms the original approach and other state-of-the-art taggers (e. g. based on HMM, see Gao & Johnson, 2008; Graça et al., 2009).

7.1.3 unsuPOS

The approach presented in Biemann (2006b) is distinct from the other regarding at least two major points. First, it does employ a graph clustering algorithm and thus, the number of induced tags can be detected by the clustering algorithm itself instead of using a predefined number of clusters. Secondly, two separate runs are performed for high / medium frequency words and for medium / low frequency words.

Similarly to the other approaches, the 200 most frequent words are used to build context vectors with a maximum distance of two. Instead of clustering all word types into fixed clusters, only the 10000 most frequent word types are clustered into fixed classes.

After merging both clusterings for high / medium and medium / low frequency words, a Viterbi model is trained. It is able to tag words that are previously unseen (all words with rank > 10000) and thus, provides disambiguation for out-of-lexicon words.

7.2 ADVANCED UNSUPERVISED POS TAGGING

Most approaches to unsupervised POS tagging ignore proper disambiguation of words. Lamar et al. (2010b) present the upper bound in accuracy of M-to-1 mapping of tagging without considering ambiguity of words and find it to be about 95% on WSJ. Regarding other tagsets, this upper bound is much lower. An

example is the very frequent word *to*. A special tag *TO* is used to tag this word veiling the word's syntactic ambiguity.

A deeper analysis shows that low frequency and previously unseen words often are disambiguated as they will not be clustered into a class directly. Tag assignment is achieved by methods relying on the context of the word (e. g. HMM as in Biemann (2006b)) instead of the word itself. This is not the case for high frequency words which bear syntactic information about the structure of a sentence.

To visualize this effect, some high frequency words are shown along with their different syntactic functions. Tables 7 and 8 (see Hänig, 2010b) give some examples for English (SUSANNE corpus[1]) and German (TIGER corpus, see Brants et al., 2002).

English						
to	Infinitive marker	63%	Preposition	37%		
that	Conjunction	75%	Pronoun	15%	Determiner	10%
as	Preposition	82%	Adverb	18%		
this	Determiner	62%	Pronoun	38%		
about	Preposition	77%	Adverb	23%		

Table 7.: Examples of different syntactic functions of high frequency words for English

These high frequency words often bear syntactical functions, so it is essential for syntactic processing of a sentence to appropriately disambiguate the different syntactic functions of the contained words. The effect on subsequent steps relying on POS tagging (e. g. syntactic parsing and relation extraction) is more distinctive than being directly observed during POS tagging evaluation. Furthermore, Lamar et al. (2010a) state that removing the disambiguation ability from a disambiguating model increases its accuracy. This is counterintuitive as this step prohibits parsers to rely on correct syntactic categories from the beginning.

1 http://www.grsampson.net/RSue.html

German				
die	Determiner	89%	Subst. rel. pronoun	10%
	Subst. dem. pronoun	1%		
das	Determiner	76%	Subst. dem. pronoun	17%
	Subst. rel. pronoun	8%		
zu	Infinitive marker	64%	Preposition	32%
	Particle	4%		
auf	Preposition	94%	Particle of sep. verb	6%
als	Preposition	61%	Comp. conjunction	31%
	Subord. conj. w. sentence	8%		

Table 8.: Examples of different syntactic functions of high frequency words for German

Sophisticated approaches to unsupervised parsing rely on POS tags in any form and depend on the tagger's accuracy and ability to assign correct – and in some cases disambiguated – syntactic roles to word types.

The high number of part-of-speech classes induced in an unsupervised manner entails difficulties for parsing approaches. The classes are more fine-grained (and sometimes consist of only a few word types) than common parts-of-speech which leads to low significance / probability values for subtrees, co-occurrences or substrings. Consequently, these syntactic dependencies may not be induced[2]. Based on the distribution of parts-of-speech over several classes, multiple similar rules have to be induced to cover certain syntactic dependencies (e. g. *A* and *NN*, *AN* and *NN* and *THE* and *NN* where *A*, *AN* and *THE* denote the classes containing *a*, *an* and *the*, respectively).

In order to improve the performance of POS taggers with regard to facilitation of subsequent processing steps, a method for disambiguation of high frequency words is presented.

2 See Section 8 for more details on unsupervised parsing methods.

7.2.1 Clustering of Contexts

Distributional similarity of word types is used to induce part-of-speech classes. In order to disambiguate different syntactic roles of a word type, the contexts the word type occurs in need to be clustered. Each of the resulting context clusters embodies a syntactic function.

From a corpus linguistic perspective, the local context $context_{NB(d)}(w_x, s)$ (see Sec. 4.2.2.1) of the xth word w_x of sentence s contains the d left and d right neighbors of w_x. If the context hits a sentence boundary, the remaining context words are filled with special tokens marking the beginning and the end of a sentence to retain this information.

The global context $context_w$ is the sum of all local contexts the word w appears in. The objective is to divide the global context into as many clusters as the number of syntactic functions the word holds. To do that, a similarity function between local contexts needs to be defined. Similarly to the similarity of two words, similarity $sim(c_a, c_b)$ between two local contexts $c(a)$ and $c(b)$ is calculated as the weighted mean of their component's similarity:

$$sim(c(a), c(b)) = \sum_i w(i) \cdot sim(c(a)_i, c(b)_i) \tag{7.2}$$

The weighting function $w(i)$ determines the weight of the ith component. Intuitively, directly preceding or succeeding words seem to be more important than more distant ones. In order to confirm this intuition, several weighting functions are studied:

UNIFORM All components of a context vector obtain the same weight – independently from the distance to the target word.

$$w(i) = \frac{1}{2n} \tag{7.3}$$

LINEAR DESCENDING Direct neighbors have more influence than distant neighbors and yield higher weights. A linear approximation is employed to decrease the weights with increasing distance.

$$w(i) = \frac{n - \left|\frac{2n-1}{2} - i\right| + \frac{3}{2}}{n \cdot (n+1)} \tag{7.4}$$

EXPONENTIAL Direct neighbors have more influence than distant neighbors and yield higher weights. An exponential approximation is employed to decrease the weights with increasing distance.

$$w(i) = \frac{2^{n-\left|\frac{2n-1}{2}-i\right|+\frac{1}{2}}}{2^{n+1}-2} \tag{7.5}$$

LINEAR ASCENDING This weighting function is proposed to prove the initial assumption. Contrary to other functions, weights linearly increase with increasing distance.

$$w(i) = \frac{\left|\frac{2n-1}{2}-i\right|+\frac{1}{2}}{n\cdot(n+1)} \tag{7.6}$$

Each of the weighting functions can be employed using different context sizes n. The respective influence is explored within the experiments.

7.2.2 Integration

The number of different syntactic roles a word type holds is a priori unknown and a well-known problem emerges: into how many clusters should the items be clustered? This is essentially a clustering problem and at this point, a graph clustering algorithm (Chinese Whispers, see Biemann, 2006a) is applied to determine the number of clusters for each word type.

The complete tagging process can be briefly described as:

```
tag corpus with existing non-disambiguating tagger

for each of the most frequent 1000 words:
  collect contexts
  calculate similarities between observed local contexts
  cluster contexts
  re-tag instances of the word with new tag
```

This integration is very flexible as every POS tagger can be extended using this approach. The re-tagging of word types can be skipped, if only one syntactic

function is found. Otherwise, all instances clustered into the dominant syntactic function will retain the original tag while all instances bearing further syntactic roles are tagged by a new tag.

7.2.3 Evaluation

Three different evaluation methods are applied. The first one evaluates the clustering performance itself, estimating the influence of parameters. The second method shows how unsupervised POS tagging can avoid clusters containing only a few word types. The third evaluation uses established measures to estimate the overall influence on the task of POS tagging.

7.2.3.1 *Context size and weighting function*

First, the clustering performance itself will be evaluated. Therefore, both corpora (SUSANNE for English and TiGer for German) are transformed so that each word type is tagged by its most frequent POS tag. This is the exact state after (perfect) non-disambiguating POS tagging. A beneficial side effect is that interpretation of the resulting clusterings is much easier compared to interpretation of unsupervised POS tags which are labeled by numbers due to lacking information about the contained part-of-speech.

The objective of this evaluation is to measure the influence of context size and applied weighting function. The *cluster purity* is calculated for each possible combination of context size n and the applied weighting function w to find the most suitable one. Cluster purity p_{c_i} of a cluster c_i is defined as

$$p_{c_i} = \frac{1}{|c_i|} max_k \left(|c_i|_{class=k} \right) \tag{7.7}$$

where $|c_i|$ denotes the cluster size of c_i and $|c_i|_{class=k}$ is the number of items of class k assigned to cluster c_i. The overall purity P of a clustering on a dataset D is the weighted sum of the individual cluster purities:

$$P = \sum_i \frac{|c_i|}{|D|} p_{c_i} \tag{7.8}$$

The impact of weighting functions depends on context size. All weighting functions show similar behavior for smaller n and diverge with increasing context size. As stated by related work on unsupervised POS tagging, the baseline is very high. Regarding cluster purity of the initial non-disambiguating tagging, the scores are 0.931 for English and 0.921 for German. These scores also confirm the assumption that ambiguity does only cover a small portion (at most 7% for English) and that English language shows less ambiguity than German (regarding to these two tagsets).

The results obtained after clustering are given in Table 9:

	uniform	linear descending	exponential	linear ascending
English				
$n = 1$	**0.932**	**0.932**	**0.932**	**0.932**
$n = 2$	0.934	**0.936**	**0.936**	0.933
$n = 3$	0.933	0.935	**0.936**	0.933
German				
$n = 1$	**0.941**	**0.941**	**0.941**	**0.941**
$n = 2$	0.938	**0.940**	**0.940**	0.926
$n = 3$	0.925	0.928	**0.929**	0.923

Table 9.: Cluster purity depending on context size and applied weighting function

The exponential weighting function achieves the best purities independent from the context size n.

A context size of 2 seems to suffice for this task as it yields the best combined purity for both languages and no other context size achieves significantly better results. *Linear descending* and *exponential* weighting functions are identical for $n \leqslant 2$ and thus, applying the computationally less complex *linear descending* function is recommended.

Another interesting aspect is that for German data, bigger context size affects the results negatively. This observation is contrary to most machine learning approaches where more data or features yield better results.

Although both baselines are very high, clustering contexts achieves an error reduction of 7.2% for English and 25.3% for German in comparison to the gold standard. In the following, the influence on parsing is depicted using an example, because cluster purity can be gamed by putting each instance into a separate class.

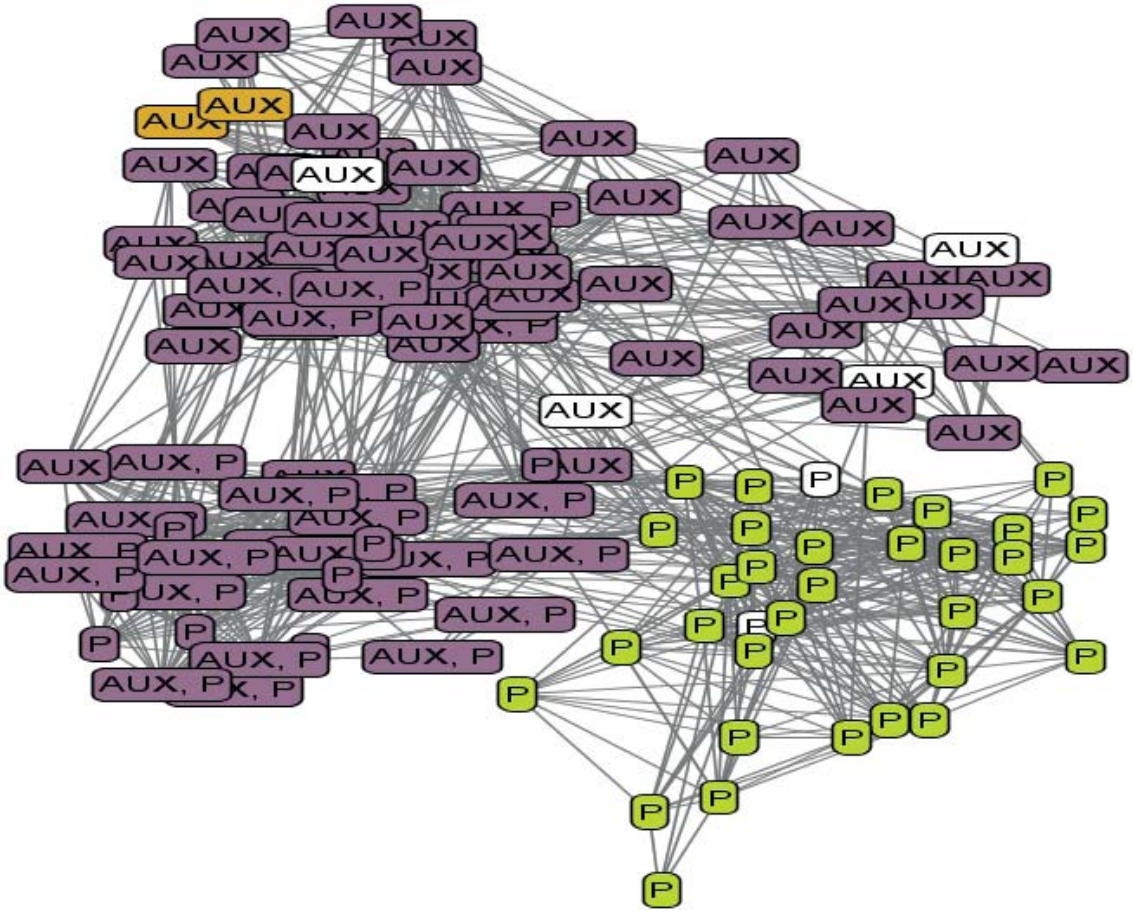


Figure 15.: Resulting clusters for *to*

The obtained clusters for *to* are shown in Figure 15. Basically, two huge clusters are detected, one contains only prepositions (labeled *P*, light-green cluster at the lower right part of the figure), the other one mainly contains particles marking infinitives (labeled *AUX*; purple cluster). The smaller cluster containing prepositions is absolutely pure, while the other one erroneously contains prepo-

sitions, too. Non-disambiguating part-of-speech tagging tags all occurrences of *to* using the same tag. Context clustering distinguishes both syntactic functions of *to* correctly. Having untangled these functions facilitates unsupervised grammar induction as statistics will be correct (or at least improved) and no syntactic function of a word type is veiled behind a more significant one. The actual degree of influence depends on the induction algorithm and ranking of different possible syntactic structures.

7.2.3.2 *Influence on small word classes*

Non-disambiguating unsupervised POS tagging often induce clusters containing only one or few word types. This originates in initial centroid detection (in the case of *SVD2*) or the fact that contexts of the word types are not similar enough to be clustered into one class. This evaluation method shows that clustering contexts can fix this problem as disambiguated subclusters show significantly higher distributional similarity to other word types with the same syntactic function.

In this experiment, five German prepositions (*ab*, *aus*, *mit*, *vor* and *zu*) are studied. All of them can also be used as parts of separable verbs. The corresponding distributional similarity scores are given in Table 16a. Most similarity scores are not very high (except for some word type pairs), hence they will end up in different word classes. After application of context clustering, the respective clusters containing prepositional instances are selected manually. Afterwards, similarity values are calculated for the extracted subclusters and as it can be seen in Table 16b, they show significantly higher similarity scores.

Unsupervisedly induced tagsets often contain a larger number of classes (approx. 300 – 500 word classes) than manually created tagsets. Incorporation of context clustering can reduce the number of induced classes naturally (e. g. instead of forcing the induction of at most k classes when using k-means clustering). Furthermore, the induced word classes can be labeled manually based on characteristic prototypes for each cluster (see Lamar et al., 2010b) if the number is not that high. This will result in semi-supervised POS tagging without huge manual effort.

	aus	mit	vor	zu
ab	0.763	0.662	0.651	0.428
aus	-	0.953	0.821	0.587
mit		-	0.810	0.603
vor			-	0.513

(a) Without context clustering

	aus	mit	vor	zu
ab	0.932	0.740	0.921	0.914
aus	-	0.957	0.993	0.977
mit		-	0.820	0.972
vor			-	0.784

(b) With context clustering

Figure 16.: Distributional similarity scores of five German prepositions

7.2.3.3 *Evaluation of POS Tagging with disambiguation*

Finally, the resulting tagging with disambiguation is evaluated using well established measures (see Sec. 4.4.4). For both languages, the approach *SVD2* presented in Lamar et al. (2010b) is initially trained on one million sentences obtained from the *Projekt Deutscher Wortschatz*[3] (see Biemann et al., 2004; Quasthoff et al., 2006). Instead of putting the evaluation corpus into the training set, the 50000 most frequent words are clustered. Similar to *unsuPOS*, a HMM is trained to finally tag the evaluation corpus including previously unseen words.

Afterwards, contexts of the 1000 most frequent word forms (ignoring punctuation) are clustered to unveil their different syntactic functions. Re-tagging is applied to those 1000 word types and a second evaluation run is performed. Table 10 gives the result of both runs for English and for German data[4].

As it can be seen, additional context clustering improves tagging accuracy for both languages independently from the applied measure. Although baselines for tagging without disambiguation are very high, proper disambiguation for high frequency words achieves a significant error reduction.

3 http://wortschatz.uni-leipzig.de/

4 For measures VI and NVI: lower scores denote better clusterings.

Model	language	M-to-1	VI	V	NVI
SVD2	en	0.58	3.63	0.61	0.83
SVD2 + context clustering	en	**0.63**	**3.21**	**0.65**	**0.73**
SVD2	de	0.82	3.20	0.60	1.07
SVD2 + context clustering	de	**0.85**	**3.08**	**0.61**	**1.03**

Table 10.: Evaluation of POS tagging without and with context clustering

7.3 POS TAGGING – CONCLUSIONS

Although disambiguation of a word's different syntactic functions is ignored by most approaches to unsupervised POS tagging, it is possible to contribute to improved POS induction.

Disambiguation was only possible for previously unseen words. Depending on the local context, out-of-lexicon words get the most probable tag assigned employing any kind of maximum likelihood estimation (e. g. HMMs). With the ability of disambiguation of high frequency words, additional potentials emerge. The accurate differentiation of a word's functions leads to superior word class purities. It is also possible to reduce the number of induced word classes in a more natural way. Two possibilities exist so far: to induce a predefined number of word classes or to use graph clustering to automatically detect the number of classes. The latter method results in a very high number of clusters, while the first possibility creates a dependency to manual tagset definition (in terms of tagset size) which is not desirable for an unsupervised method.

Further processing steps benefit from improved word classes regarding the words bearing syntactic functions. This especially holds for domain corpora as characteristic terminology occurs more frequently than in general language data (an example from the automotive domain: words like *pump* occur very often and require proper disambiguation as it can be used as noun or as verb).

There is still a great potential hidden in a closer incorporation of context clustering with an existing approach to POS tagging. Instead of incorporating context clustering as a post-processing step, it can be used to incrementally improve a model when being incorporated in an iterated algorithm like *SVD2*.

8

UNSUPERVISED CONSTITUENCY PARSING

Syntactic parsing usually consists of two steps: grammar induction and parsing as the task of grammar application. In supervised parsing, a machine learning algorithm is applied to a training set augmented with manually annotated parses. Hence, all information provided by the training corpus can be learned and reproduced during the actual parsing step. This information includes at least structure (based on constituents or dependencies) and phrase types.

Unsupervised grammar induction does not rely on the existence of annotated syntactic structures. Solely part-of-speech annotations are essential for most approaches. These can be induced in supervised or unsupervised manner (see Section 7). Some approaches operate on word forms themselves (see Seginer, 2007) while most approaches use POS tags as abstraction layer and to reduce data-sparseness to an acceptable level. Klein & Manning (2002a, 2004) state that the extension of grammar induction methods to induce trees from words instead of from POS tags is rather straightforward as high-accuracy unsupervised taggers exist (see Bod, 2006b).

In the following, some state-of-the-art unsupervised parsers are presented. Potential for improvements with special regard to RE is estimated and an approach incorporating all of these improvements will be developed.

8.1 UNSUPERVISED PARSING – RELATED WORK

8.1.1 Constituent-Context Model

The Constituent-Context Model (CCM) is fundamentally based on the assumption that constituents appear within constituent contexts (see Klein & Manning, 2002a). This is a simplified version of classic linguistic constituency tests (see Radford, 1988). The phenomenon being exploited is that long constituents frequently have shorter equivalents or proforms occurring in similar contexts. The CCM is designed to detect easily discoverable constituents and to encode constituency directly into a sequence's context. Sequences occurring in that context will be parsed as constituents during the next iterations.

CCM describes contiguous subsequences including empty ones. A span α encloses a terminal sequence such as *DT JJ NN*. Each span occurs in a local context c (e. g. ∘ – *VBZ*) where c is an ordered pair consisting of preceding and following terminals[1]. A bracketing of a sentence can be presented as a boolean matrix B which indicates which sequences are constituents and which are distituents.

The generative model has two phases:

PROBABILITY ASSIGNMENT FOR SENTENCES A bracketing B is chosen according to a distribution $P(B)$ and a sentence S is generated with:

$$P(S, B) = P(B)\, P(S|B) \tag{8.1}$$

Context and terminal sequence of a span are independent of each other, and thus:

$$P(S|B) = \prod_{\langle i,j\rangle \in spans(S)} P\left(\alpha_{ij}, x_{ij}|B_{ij}\right) = \prod_{\langle i,j\rangle \in spans(S)} P\left(\alpha_{ij}|B_{ij}\right) P\left(x_{ij}|B_{ij}\right) \tag{8.2}$$

where $P\left(\alpha_{ij}|B_{ij}\right)$ is a pair of multinomial distributions – for constituents and distituents, $B_{ij} = c$ and $B_{ij} = d$, respectively[2].

1 ∘ denotes a sentence boundary

2 $P\left(x_{ij}|B_{ij}\right)$ analogously for contexts

The probability of a sentence S is defined as the sum of all possible bracketings for this sentence as in:

$$P(S) = \sum_{B} P(B)\, P(S|B) \tag{8.3}$$

STRUCTURE INDUCTION In order to run entropy maximization over this model, all sentences S are treated as observed and the corresponding binary bracketings B as unobserved. The yield and context distributions $P(\alpha|b)$ and $P(x_{ij}|b)$ form the parameters Θ. During the E-Step, the completion likelihoods $P(B|S,\Theta)$ are calculated for the current Θ. During the M-Step, Θ' will be detected, so that $\sum_B P(B|S,\Theta) \log P(S,B|\Theta')$ is maximized.

The original Constituent-Context Model has been extended by incorporation of an additional dependency model – the Dependency Model with Valence (DMV) (see Klein & Manning, 2004; Klein, 2005). The combination of CCM and DMV achieves superior results.

8.1.2 Data-Oriented Parsing

A generalization of the CCM is presented by Bod (2006a,b). The all-subtrees approach operates on a richer feature space than other models that study contiguous substrings only (see van Zaanen, 2000). Data-Oriented Parsing (DOP) originally has been presented by Bod (1998) as a supervised parsing approach exploring the subtrees of an annotated corpus. In more recent research, this approach is generalized to be applicable to the task of unsupervised parsing. The basic idea is to generate all possible trees for each sentence of the training corpus and use their frequencies to estimate the probability of certain parse trees. Only unlabeled binary trees are taken into account. A deeper study of DOP shows that various linguistic phenomena can be learned by an all-subtree approach, e. g. such as agreement and movement (see Bod, 2007a).

Unsupervised Data-Oriented Parsing (U-DOP) basically consists of three steps:

ASSIGN ALL POSSIBLE TREES TO A CORPUS All possible binary trees are generated for a given sentence's POS sequence. An example is given in Figure 17 (see Bod, 2006b).

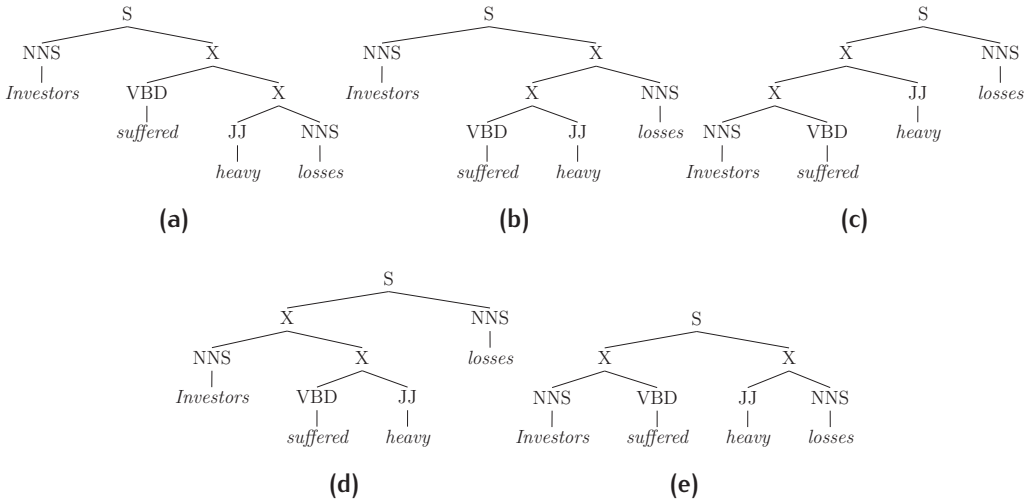


Figure 17.: Possible binary trees for *Investors suffered heavy losses*

Internal nodes are labeled *X*. As the number of parse trees grows rapidly with the length of a sentence, for sentences containing 7 or more words only a random sample of all possible trees will generated (ranging from 60% to 7.5% of all possible trees – decreasing for longer sentences).

TREE CONVERSION All subtrees from the binary trees are employed to calculate the most probable parse tree for a given tag sequence. Some example subtrees for the sentence given in Fig. 17 are shown in Fig. 18.

Contrary to most other approaches, discontiguous sequences are taken into account (e. g. *Investors* ... *losses* in Fig. 18b). As in the original DOP approach, subtrees are combined using a leftmost node substitution operation $\circ$ to build parse trees out of a variety of subtrees.

COMPUTE THE MOST LIKELY TREE FOR A SENTENCE The probability of a subtree t is defined as the ratio of the subtree's frequency and the frequency of all subtrees with the same root label (see Equ. 8.4):

$$P(t) = \frac{|t|}{\sum_{t':r(t')=r(t)} |t'|} \tag{8.4}$$

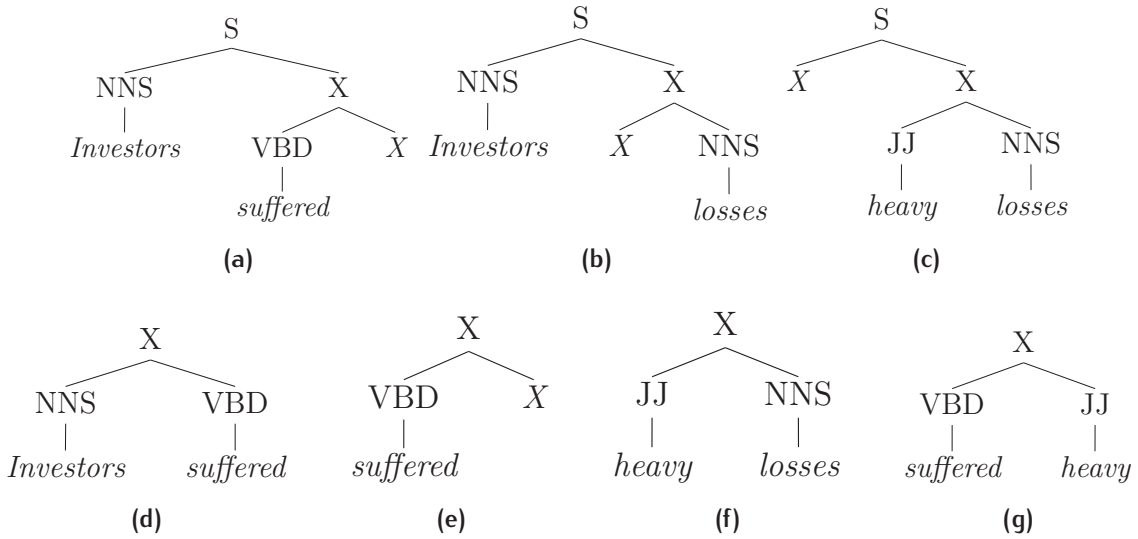


Figure 18.: Some subtrees for *Investors suffered heavy losses*

The probability of a derivation is calculated as the product of its subtree's smoothed probabilities:

$$P(t_1 \circ \ldots \circ t_n) = \prod_i P(t_i) \tag{8.5}$$

The same parse tree can be generated with different derivations and thus, the probability of a certain tree is the sum of the probabilities of all possible derivations. The probability of a tree is given in Equ. 8.6 where t_{id} denotes the i-th subtree of derivation d.

$$P(T) = \sum_d \prod_i P(t_{id}) \tag{8.6}$$

Instead of using frequencies of the subtrees for probability estimation, subtree parameters also can be learned by entropy maximization (UML-DOP, see Bod, 2007b).

8.1.3 Common Cover Links

The approach presented by Seginer (2007) differs from the other ones in multiple points.

First, it does not rely on POS tags in either supervised or unsupervised form, as the algorithm itself collects labels containing contextual information. These labels represent sort of word classes which are directly used for parsing.

Secondly, this approach does not operate on trees directly, instead it assigns links between words reminding of dependencies. The induced Common Cover Links (CCL) are converted into bracketings in a subsequent step.

And thirdly – inspired by human language processing (see Crocker et al., 1999) – it parses incrementally.

A *common cover link* over a sentence X is defined as a triple $x \xrightarrow{d} y$ where x and y are words and d is a non-negative integer. The word x denotes the base of this link, y denotes the head. The depth d of the link is the depth of word x so that d is the maximal number of brackets $x \in X_1 \subset \ldots \subset X_n \subset \mathcal{B}$ (X_i are brackets contained in the bracketing $\mathcal{B}$ of a sentence). An example is given in Fig. 19 – the constituency tree of *I know the boy sleeps* is given in Fig. 19a and the equivalent common cover links representation is given in Fig. 19b:

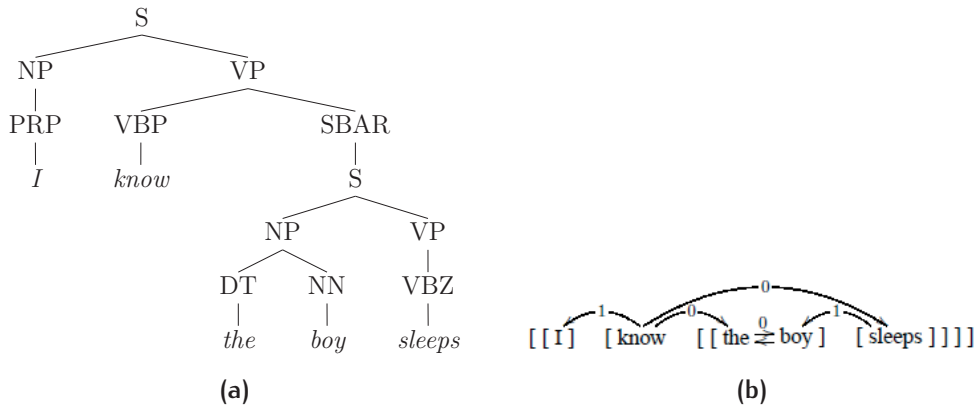


Figure 19.: Constituency tree and common cover link representation of *I know the boy sleeps*

An incremental parser is used to compute shortest common cover links for each sentence. Incremental parsing means that a sentence is read word by word and the parser is only allowed to add links ending at the last word as it does not know any further words. A non-negative lexicalized *weight frunction* assigns weights to link candidates which may be added between a word from the already known prefix $\langle x_1, \ldots, x_{k-1} \rangle$ and the current word x_k.

The lexicon contains a lexical entry for each word x. Each entry consists of adjacency points representing counts of neighboring words and a special set of *linking properties* $P = \{\mathrm{Stop}, \mathrm{In}*, \mathrm{In}, \mathrm{Out}\}$. These linking properties specify the strength for non-attachment, inbound and outbound connections. These properties are similar to parts-of-speech as they provide contextual information.

During each step, the best possible link is attached and the lexicon is updated iteratively improving statistics during parsing.

8.1.4 Conclusions

The presented algorithms achieve state-of-the-art results for grammar induction. All approaches solely induce constituency trees.

With respect to information extraction, additional knowledge is desirable:

HEAD AND MODIFIER DETECTION Information extraction often targets certain entities. These entities often occur within texts as nominal expressions or pronouns. As unsupervised POS tagging is not able to assign appropriate tags (e. g. *NN* or *JJ*) to distinguish between attributes and entity candidates, this could be done by a parser exploring this kind of dependency.

PHRASE LABELING Unsupervised parsers often induce unlabeled parse trees. For information extraction, it is useful to know, which phrases contain similar information. Especially when employing unsupervised POS tagging, there is no single word cluster containing all nouns or proper nouns - instead there multiple ones containing (proper) nouns. Extraction of nominal or prepositional phrases needs appropriately assigned phrase types which are not supported by the presented grammar induction methods.

OPTIONAL SEMI-SUPERVISION In real-world applications, the accuracy of results has the highest priority. Thus, a desirable feature is a convenient possibility for manual adaptation. Thus, the parser should be easily extendable by human intervention. It should be possible to alter the induced grammar without huge effort and to label phrases in a human-readable

way. Complex probabilistic models operating on strings or trees do not provide this opportunity.

8.2 EVALUATION OF CONSTITUENCY PARSERS

Evaluation of grammar induction algorithms is a difficult task as the objective is not as obvious as it may appear. Klein (2005) basically distinguishes between three possible goals of grammar induction:

1. The production of a probabilistic language model,
2. the annotation of sentences to facilitate further processing, and
3. the automation of the job of a linguist.

As the objective of this thesis is not linguistically motivated, argument (3) is out of scope. Although the question "How do humans learn language?" is of major interest and influences research especially for unsupervised algorithms. However, some ideas were integrated in the algorithms described in this work.

In this thesis, unsupervised parsing is used to augment textual data with syntactic features for further processing like Relation Extraction. Thus, the induced grammar itself is of minor interest (point (1)). Evaluation of parses for further processing can be achieved in two ways (argument (2)).

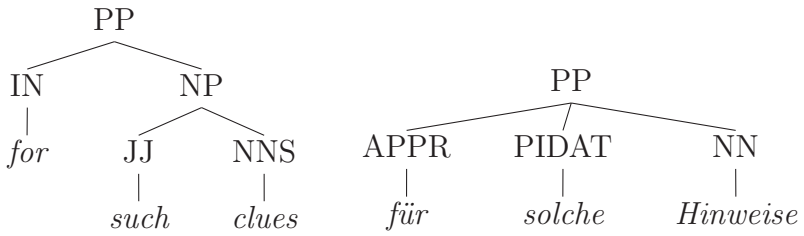
First, a task-oriented evaluation directly measures the influence on the actual approach. While different tasks rely on different prerequisites a general parser evaluation is not able to ensure best quality for all tasks. One criterion is the deepness of induced parses. Most state-of-the-art parsers produce deep and detailed parses, but it has been shown for some tasks (such as Relation Extraction) that shallow parses suffice (see Zelenko et al., 2003).

This leads to the second evaluation method: comparison of the produced parse trees and an appropriate gold standard. Measuring agreement between an induced parse tree and the corresponding gold parse can be done in a very

convenient way (see Sec. 8.2.1). But there is at least one drawback: the dependency on the gold standard. While linguists argue about the *true* structure of language several treebanks were created representing the current – or at least a widely accepted – opinion of how correct structures should look like.

Two corpora are well established for parser evaluation: the Wall Street Journal section of the Penn Treebank (see Sec. 5.3.1) for English and the NEGRA Corpus (see Sec. 5.3.2) for German. Even these two gold standards show substantial differences and thus, a parser could perform well on one of them and bad on the other just because of inconsistencies within the evaluation data.

It is noticeable that both corpora show significant disagreements of the depth of parse trees. This effect can be observed even for high frequent constructions like pronoun phrases:



(a) WSJ Corpus **(b)** NEGRA Corpus

Figure 20.: PPs in the WSJ Corpus and the NEGRA Corpus

The WSJ is annotated by much deeper structures than the NEGRA is. Fig. 21 gives an overview about average parsetree depths of the annotated sentences in both corpora. While the parse depth of NEGRA sentences seems to converge to 7 or 8, no upper limit can be assumed by the graph for the WSJ.

8.2.1 Unlabeled Brackets Measure

Comparing induced parse trees to gold standard trees is a better understood way than comparing grammars. Furthermore, it is possible to compare parsers inducing different grammar types as only the resulting syntactic structure is of interest.

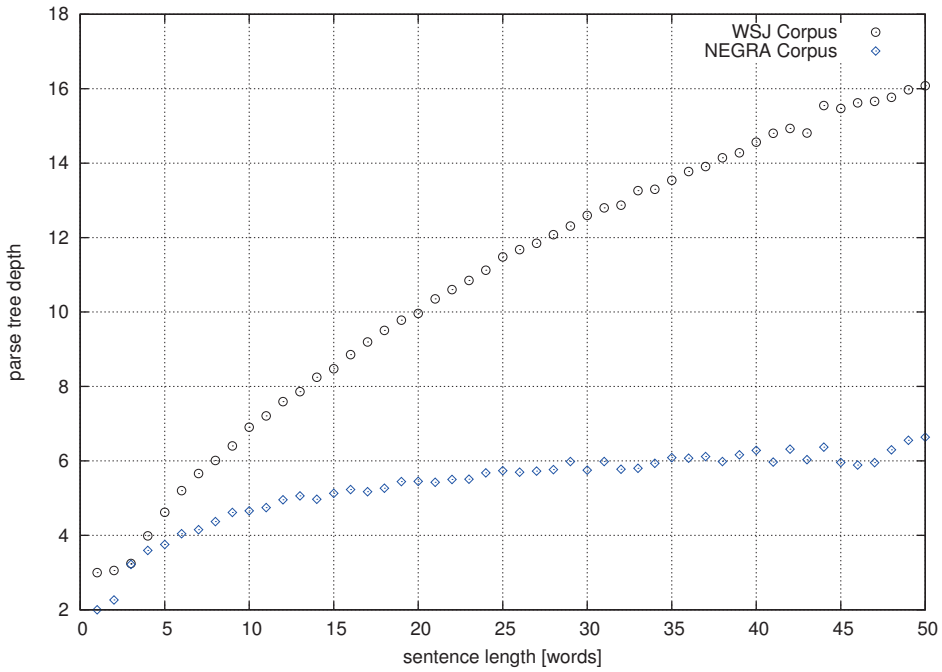


Figure 21.: Average parse tree depth

Measures based on *PARSEVAL* (see Black et al., 1991; Grishman et al., 1992) are well established. They calculate precision and recall scores for brackets and thus, reduce the tree comparison task to a simple Information Retrieval task on bracketings. Nevertheless, several disadvantages exist (see Carroll et al., 1999) and will be discussed later.

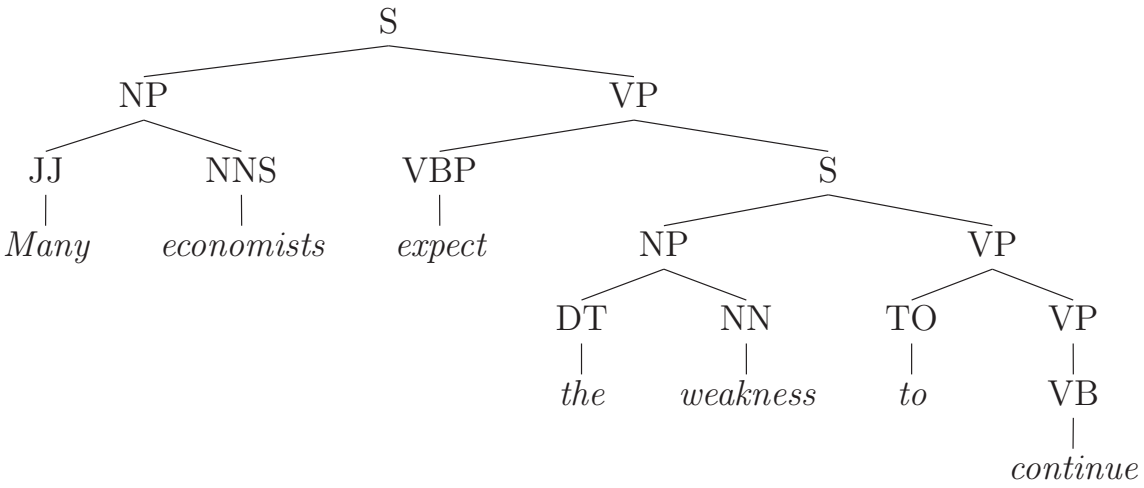
While *PARSEVAL* measures can be applied to both supervised and unsupervised parsers, labels attached to phrases embody the discriminating criterion. Supervised parses contain additional information beneath the bare tree structure which can be used for a comprehensive measure (e. g. constituent labels as proposed by Collins (1996)). As unsupervised algorithms cannot assign phrase labels as given in the respective gold treebank, Klein (2005) proposes the usage of the Unlabeled Brackets Measure (UBM).

Each labeled parse tree T can be represented as set of labeled constituent brackets $(x : i, j)$, where x is the label of the corresponding node n, i is the index of the left material dominated by n and analogous, j is the right index. All nodes

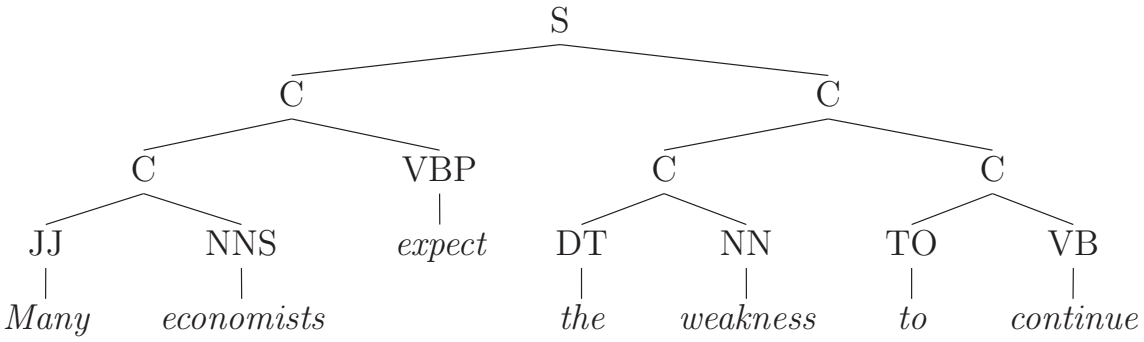
containing exactly one leaf node are ignored. This is the case for *terminal* and *preterminal* nodes as well as for nodes dominating only one terminal (e. g. VP as in VP – VB – *continue*, see Fig. 22(a)).

An example is given by a gold sentence from the WSJ Corpus (see Fig. 22 for the corresponding parse trees):

$_0$ Many $_1$ economists $_2$ expect $_3$ the $_4$ weakness $_5$ to $_6$ continue $_7$



(a) Gold Tree



(b) Predicted Tree

Figure 22.: A gold tree and a predicted tree for the sentence "Many economists expect the weakness to continue"

All six valid brackets representing the gold tree are given in Table 11.

A set of unlabeled brackets can be derived from this labeled brackets set as given in Equ. 8.7:

$$\text{brackets}(T) = \{\langle i, j\rangle : \exists x \text{ s. t. } (x : i, j) \in T\} \tag{8.7}$$

Constituent	Material Spanned
$(S : 0, 7)$	Many economists expect the weakness to continue
$(NP : 0, 2)$	Many economists
$(VP : 2, 7)$	expect the weakness to continue
$(S : 3, 7)$	the weakness to continue
$(NP : 3, 5)$	the weakness
$(VP : 5, 7)$	to continue

Table 11.: Labeled brackets of the gold tree in Fig. 22(a)

To compare a corpus P containing the predicted parse trees P_i against a gold standard corpus G containg the gold trees G_i precision and recall scores need to be defined. For unlabeled parse trees, the definition of unlabeled precision (UP) and unlabeled recall (UR) are given in equ. 8.8 and 8.9, respectively.

$$\text{UP}(P, G) = \frac{\sum_i |\text{brackets}(P_i) \cap \text{brackets}(G_i)|}{\sum_i |\text{brackets}(P_i)|} \tag{8.8}$$

$$\text{UR}(P, G) = \frac{\sum_i |\text{brackets}(P_i) \cap \text{brackets}(G_i)|}{\sum_i |\text{brackets}(G_i)|} \tag{8.9}$$

The F-score (harmonic mean of UP and UR) is defined by:

$$\text{UF}(P, G) = \frac{2 \cdot \text{UP}(P, G) \cdot \text{UR}(P, G)}{\text{UP}(P, G) + \text{UR}(P, G)} \tag{8.10}$$

For the example given in Fig. 22 the predicted tree contains one incorrect bracket ($\langle 0, 3 \rangle$: "Many economists expect"). The bracket $\langle 2, 7 \rangle$ ("expect the weakness to continue") is missing. Hence, UP, UR and consequently UF each achieve a score of $\frac{5}{6}$.

The Unlabeled Brackets Measure differs from the standard PARSEVAL in several points: brackets of length one and multiple brackets over the same span are ignored as well as phrase labels.

Although having the possibility to calculate objective scores for evaluation of parsers is very convenient, neither the Unlabeled Brackets Measure nor the original PARSEVAL provide *reliable* scores due to several problems. Both measures penalize parsers predicting to many brackets – compared to the gold standard – even if they are correct (see Srinivas et al., 1996). Furthermore, correct phrase boundary detection does not guarantee correct semantic reading (see Lin, 1996).

Measures based on PARSEVAL have one minor disadvantage: the parser has to produce constituency parse trees. To circumvent this problem, produced parses can also be transformed into constituency parses (for transformation of dependency parses, see Xia & Palmer, 2001).

8.2.2 Constituent Chunk Score

The Constituent Chunk Score (CCS) measures the correct number of chunks extracted by a parser as in Ponvert et al. (2011). In this context *chunks* denote unlabeled non-overlapping multiword constituents. The purpose of this measure is to evaluate local constituent structures focusing on lower branches. Amongst those lower branches are noun phrases and prepositional phrases which yield valuable information for further tasks (e. g. Relation Extraction).

Following Ponvert et al. (2011), *constituent chunks* denotes the subset of gold standard constituents containing only chunks that

- consist of more than one token (branching) and
- are non-hierarchical (do not contain subconstituents).

As for the UBM, a set of unlabeled brackets containing constituent chunks can be defined:

$$\begin{aligned} c-chunks(T) = \{\langle i,j\rangle : \exists x \text{ s. t. } (x:i,j) \in T \\ \wedge\, i+1 < j \\ \wedge\, is-non-hierarchical(\langle i,j\rangle)\} \end{aligned} \tag{8.11}$$

Similar to the UBM, precision *CC-P*, recall *CC-R* and f-measure *CC-F* can be defined over these constituent chunks.

8.3 HOW TO BUILD A PARSE TREE?

Building parse trees is not as trivial as it may appear. Several decisions have to be made starting with a very basic one: should we use bottom-up parsing or top-down parsing? *LL parsers* (see Li, 1996) are very prominent amongst top-down parsers. Given a grammar, LL parsers process the sentence from left to right producing the leftmost derivation. Although this matches the opinion of many linguists assuming that humans process language incrementally (see Crocker et al., 1999), top-down parsing has its drawbacks. Local ambiguities force the algorithm to backtrack even though there is no evidence that people do (see Abney, 1989). The example in Fig. 23 shows such a case. At this point, the parser has to expand the VP, but without further information, the parser cannot be certain how to attach the PP correctly.

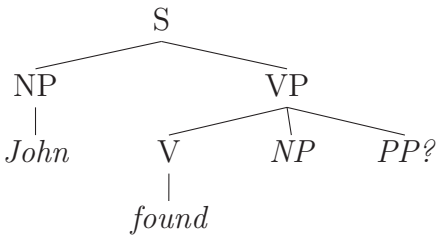


Figure 23.: Local disambiguity for PP attachment

Bottom-up parsing connects words to simple phrases and complex structures out of those resulting phrases. Starting with words and associated part-of-speech tags, higher-order structures are inferred as shown in Fig. 24.

For unsupervised grammar induction bottom-up parsing is the paradigm to choose as it does not need an existing grammar. Furthermore, it is its task to induce a grammar for a given language.

8.3.1 Greedy Learning vs. Maximum Likelihood Estimation

Basicly, two different approaches are established in the field of unsupervised grammar induction. Several approaches apply *greedy algorithms* (see Seginer,

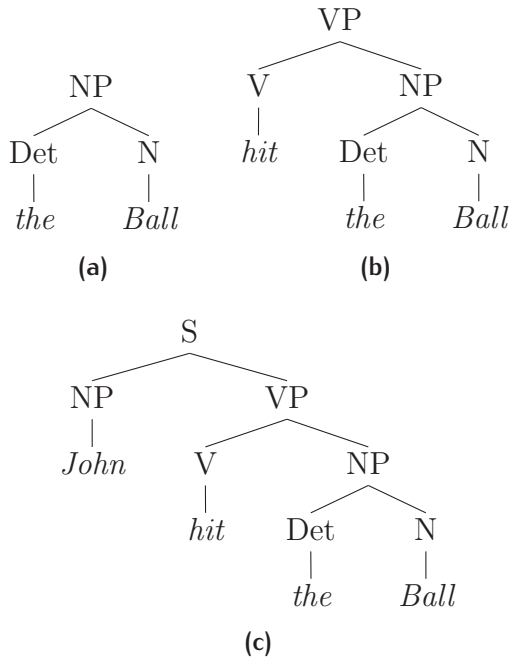


Figure 24.: Bottom-up parse tree creation for "John hit the ball"

2007; Hänig et al., 2008; Hänig, 2010a; Ponvert et al., 2011). "A greedy algorithm always makes the choice that looks best at the moment." (see Cormen et al., 2001) It makes locally optimal choices, but this does not guarantee a globally optimal solution. For many problems, finding the locally best choice suffices and leads to satisfying results.

With regard to the task of building a parse tree for a given sentence, greedy algorithms suffice. Using the bottom-up approach, the most common decision that has to be made is: which nodes should be connected to form a more complex phrase? Obviously, these are local decisions as long as contextual knowledge about the complete sentence can be neglected. Exceptions are non-contiguous constructions such as *nearest . . . to . . .* (see Bod, 2007b) which need the inclusion of some kind of discontiguous context into the induction algorithm.

Maximum Likelihood Estimation (MLE) has its origin in statistics and aims to estimate the optimal parameters for a given model (see Edwards, 1972). Given a couple of observations, MLE finds the parameters, that achieve the most likely solution.

The benefits of MLE algorithms are – in most cases – better solutions. On the other side, greedy algorithms occupy fewer resources in terms of time and hardware requirements. Additionally, greedy algorithm can produce parses on streamed data without any knowledge about the beginning or the end of sentences.

In this work, an *iterated greedy algorithm* for unsupervised parsing will be presented. The motivation behind iterative greedy learning is to retain the advantages of greedy algorithms (fast processing) and to combine them with global optimal solutions. Several locally optimal decisions are made for all data points within a given data set. Those decisions are aggregated throughout the complete data set and the most frequent one for a given situation is considered to be the best one in this case. Afterwards, the newly learned solution is applied to all corresponding data points. This process is iterated until a solution for all situations has been learned.

8.3.2 Branching

Branching is the tendency of words or smaller grammatical units towards a certain order within a sentence. Beneath languages with free word order (e. g. Czech or Warlpiri), two major manifestations can be observed: right branching languages (e. g. English, Spanish, Arabic and German) and left branching languages (e. g. Japanese, Turkish and Tamil, see Haspelmath et al., 2005).

Branching can be regarded as tendency. While several languages show strictly left or right branching, others show exceptions. English and German – although regarded as right branching languages – place prepositions, adjectives of locations and numerals in front of nouns.

Besides the sense of tendentially orientation of grammatical units, branching yields another meaning. In graph theory, the *branching factor* describes the number of children of a parent node. As constituents can (theoretically) consist of an infinite number of words, the maximum branching factor of parse trees should be ∞.

Nevertheless, most approaches to unsupervised grammar induction limit the branching factor of the produced trees to 2 in order to induce binary trees (see Bod, 2006a,b, 2007b; Klein & Manning, 2002a, 2004). The major motivation behind this is the widely accepted X-Bar Theory (see Jackendoff, 1977) which proposed the consequent usage of binary trees. Another explanation is the tendency towards dependency parsing (see Dependency Grammar, Section 4.2.4.4). Dependencies are binary relations between a head and a dependent which can be transformed into a binary dependency parse tree.

For some constituents it is more convenient to allow a branching factor bigger than two. For tasks like Entity Detection or Relation Extraction, noun phrases and prepositional phrases are of major interest and often consist of more than two tokens (e. g. *DT JJ NN*).

8.4 CONSTITUENT DETECTION

The crucial step for constituency parsing is the correct detection of constituents. Having detected as many constituents of a sentence as possible simplifies the assignment of the correct hierarchical structure of those constituents.

Obviously, there are various linguistic features and properties that can be used to derive knowledge about constituency. The following three qualify to a special degree due to their possibility of precise formalization:

SYNTAGMATIC RELATIONS Words occurring within the context of each other are considered to be syntagmatically related (see Sec. 4.2.2.2). Employing neighborhood co-occurrences the strength of such a relation between two words w_0 and w_1 can be described by the corresponding significance value $sig(w_0, w_1)$.

A high value of $sig(w_0, w_1)$ implies a strong syntagmatic relation and thus, the words w_0 and w_1 most likely belong to the same constituent.

MOVEMENT TEST As constituents can occupy different positions within a sentence, two special cases can be used directly to induce knowledge about the orientation of words *within* constituents: neighborhood co-occurrences of a given word w with the beginning and the end of a sentence, $\mathrm{sig}(\langle \mathrm{BOS} \rangle, w)$ and $\mathrm{sig}(w, \langle \mathrm{EOS} \rangle)$, respectively.

The hypothesis can be stated as: words occurring significantly often at the beginning of sentences prefer the first position within a constituent and words that occur at the end of a sentence prefer the last position of constituents.

REPLACEMENT TEST Constituents can be replaced by constituents of the same type or pronouns. A proper reformulation is that two word sequences $W = \langle v_0, \ldots, v_{n-1} \rangle$ and $V = \langle w_0, \ldots, w_{n-1} \rangle$ occurring in the same global context ($\mathrm{Context}(V) = \mathrm{Context}(W)$) stand in paradigmatic relation (see Sec. 4.2.2.3). Those constituents tend to have of the same type (see Klein & Manning, 2002a).

8.4.1 Preferred Positions

For each word two probabilities are defined. One denotes the probability of word w occurring at the left boundary of a constituent assuming that w occurs on one of the boundaries (see equ. 8.12). The other one is for the right boundary (see equ. 8.13).

$$\mathrm{pref}_{\mathrm{left}}(w) = \frac{\mathrm{sig}(\langle \mathrm{BOS} \rangle, w)}{\mathrm{sig}(\langle \mathrm{BOS} \rangle, w) + \mathrm{sig}(w, \langle \mathrm{EOS} \rangle)} \tag{8.12}$$

$$\mathrm{pref}_{\mathrm{right}}(w) = \frac{\mathrm{sig}(w, \langle \mathrm{EOS} \rangle)}{\mathrm{sig}(\langle \mathrm{BOS} \rangle, w) + \mathrm{sig}(w, \langle \mathrm{EOS} \rangle)} \tag{8.13}$$

POS tags showing strong preference towards one boundary are given for English and German in Table 12. Those extracted tags match the ones extracted by other approaches as in Santamaría & Araujo (2010) and Ponvert et al. (2011).

English				German			
Left boundary		Right boundary		Left boundary		Right boundary	
WP$	1.000	POS	1.000	KOUI	1.000	NNE	1.000
PDT	1.000	RP	0.989	VVIMP	1.000	VMPP	1.000
WDT	0.989	NN	0.879	VAIMP	1.000	VMINF	1.000
CC	0.982	NNS	0.841	KOUS	0.999	VAINF	0.995
WP	0.981	NNPS	0.810	PRELS	0.998	VAPP	0.990
WRB	0.975	CD	0.791	PWAV	0.994	VVIZU	0.990
DT	0.957	VB	0.734	PWS	0.993	PTKVZ	0.990
PRP$	0.940	RBR	0.731	PTKA	0.988	VVINF	0.982
IN	0.913	UH	0.709	PRELAT	0.986	APZR	0.977
PRP	0.872	JJR	0.638	PWAT	0.979	APPO	0.967

Table 12.: Preferred positions of POS tags for English and German

A strict version of estimating the preferred position of a tag is achieved by introduction of a threshold θ as in:

$$\text{pref}^{\theta}_{(\text{left}|\text{right})}(w) = \begin{cases} \text{pref}_{(\text{left}|\text{right})}(w) & \text{pref}_{(\text{left}|\text{right})}(w) \geqslant \theta \\ 0 & else \end{cases} \tag{8.14}$$

Incorporation of this threshold forbids induction of constituents with uncertain constituent boundaries.

8.4.1.1 *Words with atypical preferences*

Generalization to the part-of-speech level leads to ignorance of word-specific properties. Several words prefer none or the opposite boundary of its assigned part-of-speech. The detection of those words is achieved in a straight forward manner. Preferred positions are calculated on both – word forms and POS tags. Tokens having at least a minimum support of σ are considered to occur with a

statistically reliable frequency and thus, words showing the opposite preferred position as its associated part-of-speech will be treated as exceptions.

8.4.2 Candidate Scoring

To induce correct phrase structures, scores have to be assigned to constituent candidates. These scores will be used to determine the best candidate ranking for a greedy learning algorithm. Additionally, a normalized version of these scores will be used as probabilities for an algorithm based on maximum likelihood estimation.

The following propositions of a proper score cscore for a given constituent candidate $c = \langle w_0, \ldots, w_{n-1} \rangle$ will be discussed and evaluated separately.

The first score equals the significance value of c (see equ. 8.15) following the argumentation given for syntagmatic relations. Furthermore, evaluation in (Hänig et al., 2008) shows sufficency of pure significance values as scoring criterion. Especially during the first iterations of a greedy induction algorithm, log-likelihood proved to yield accurate constituents.

$$\text{cscore}_{\text{Sig}}(c) = \text{sig}(\langle w_0, \ldots, w_{n-2} \rangle, \langle w_1, \ldots, w_{n-1} \rangle) \tag{8.15}$$

Taking preferred positions of words or POS tags into account leads directly to another simple score (see equ. 8.16). $\text{cscore}_{\text{PrefPos}}$ relies solely on $c's$ first and last component and the degree to which they prefer the respective boundary.

$$\text{cscore}_{\text{PrefPos}}(c) = \text{pref}_{\text{left}}(w_0) * \text{pref}_{\text{right}}(w_{n-1}) \tag{8.16}$$

An extension to the preceding score is given in equ. 8.17. $\text{cscore}_{\text{PrefPosContext}}$ additionally takes the preferences of $c's$ preceding (w_{-1}) and succeeding (w_n) tokens into account.

$$\text{cscore}_{\text{PrefPosContext}}(c) = \text{cscore}_{\text{PrefPos}}(c) * \text{pref}_{\text{right}}(w_{-1}) * \text{pref}_{\text{left}}(w_n) \tag{8.17}$$

The last two scores describe combined models considering both, significance of candidate c along with preferred positions of involved tokens. The combina-

tion of $\text{cscore}_{\text{Sig}}$ and $\text{cscore}_{\text{PrefPos}}$ is given in Equ. 8.18. Analogously, see Equ. 8.19 for the joint model of $\text{cscore}_{\text{Sig}}$ and $\text{cscore}_{\text{PrefPosContext}}$.

$$\text{cscore}_{\text{Sig+PrefPos}}(c) = \text{cscore}_{\text{Sig}}(c) * \text{cscore}_{\text{PrefPos}}(c) \tag{8.18}$$

$$\text{cscore}_{\text{Sig+PrefPosContext}}(c) = \text{cscore}_{\text{Sig}}(c) * \text{cscore}_{\text{PrefPosContext}}(c) \tag{8.19}$$

8.4.3 Candidate Selection

Besides constituent scoring functions, corpus size and maximum allowed branching factor influence the extracted candidates.

As stated by Hänig et al. (2008), $\text{cscore}_{\text{Sig}}$ seem to suffice although a hybrid model achieves slightly better results. Among the proposed scoring functions, $\text{cscore}_{\text{Sig+PrefPosContext}}$ achieves the best results. Although the context of constituent candidates is taken into account, the induced grammar is still context-free as created grammar rules do not use contextual information.

Contrary to other approaches, the maximum branching factor is unrestricted. This is motivated by noun phrases and prepositional phrases which often consist of more than two words. All phrases could by transformed into a binary structure, but this would delegate structural analysis tasks to succeeding tasks. Induction of flat phrases containing the relevant information for Relation Extraction is most desirable at this point. Furthermore, the correct structure of coordinating conjunctions is an influential point for this decision. As proposed in Hänig (2010a), symmetric constructions containing two heads seem to be more appropriate for some languages. A restriction to binary representation forbids those analyses.

Corpus size depends on the level of language the parsing algorithm is applied to. Inducing structure on the part-of-speech level is much more generalized than on words themselves. Thus, small corpora (consisting of about 100k sentences) suffice for grammar induction on POS tags. Regarding induction on word level,

the corpus should contain at least 10 million sentences to reduce data sparseness to show only minimal negative effects.

8.5 PHRASE LABELING

Syntactic trees do not only add structure to a sentence or word sequence, they also should provide information about the type of the present phrases. This is the case for supervised parsers which rely on annotated data providing these details. Common phrase types for English (e. g. *noun phrase (NP)*, *prepositional phrase (PP)* or *verb phrase (VP)*) are attached to the root node of each phrase (see Figure 25a). Unsupervised algorithms are not able to distinguish between different phrase types like *NP* or *PP* as they cannot conceive the syntactic meaning of *noun phrase* defined by the name itself. Furthermore, unsupervised induction of parts-of-speech do not yield any hint about the syntactic roles of the word classes. As described in Section 7, the classes are labeled by numbers (see Figure 25b).

Beneath the obvious benefits of self-explanatory phrase labels, the ability to detect similar phrase types contributes to iterative greedy induction of syntactic structures (see evaluation in Section 8.6). Thus, three different label phrase type assignment methods are established.

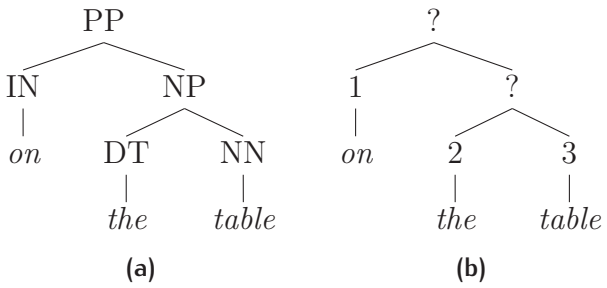


Figure 25.: Phrase labels for *on the table*

8.5.1 Enumeration of phrase types

The default method is to consecutively enumerate the induced phrase types (see Figure 26b). This leads to individual treatment of each phrase type.

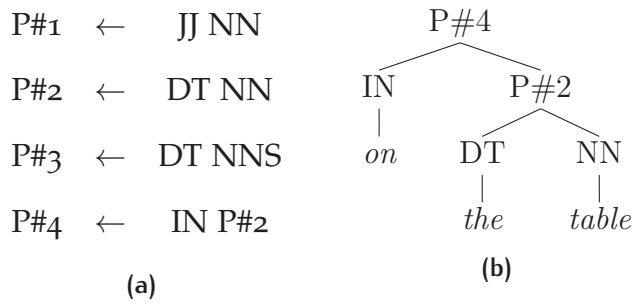


Figure 26.: Enumerated Phrase Types

Obviously, there are several syntactic properties which can be derived from the given examples:

The first example given in Figure 26a shows the creation of phrase type *P#1* which consists of an adjective and its dominating normal noun. This whole construction shows the same syntactic properties as the noun itself.

(a) *Pigs* love *tasty truffles*.

(b) *Pigs* love *truffles*.

(c) *Young pigs* love *truffles*.

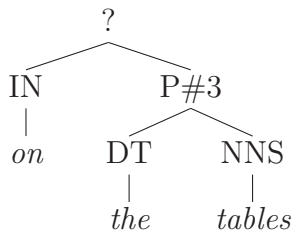
(d) *Young pigs* love *tasty truffles*.

All four examples (a) - (d) keep their grammaticality if adjectives are inserted or removed in front of nouns[3]. This observation leads to the hypothesis that the correct phrase type is at least similar to normal nouns. The expectation of a parser to induce a rule like *NP* ← *JJ NN* is raised. But as mentioned before unsupervised algorithms cannot derive phrase labels like *NP*. Instead, it can

3 Nevertheless, agreement between the different parts of the noun phrase is still required.

produce the derivation *NN* ← *JJ NN* (see Section 8.5.2) retaining information that adjectives do not change syntactic behavior of normal nouns.

Phrase types *P#2* and *P#3* in Figure 26a obviously are of the same type – noun phrases. Both constructions are constituted of a determiner and a normal noun, they only differ in grammatical number of the noun. To assign dissimilar labels to those phrases is not wrong at all, but it entails the necessity to learn identical grammar rules twice. Phrase *P#4* stands for prepositional phrases which consist of a preposition followed by a noun phrase. As there are different labels for noun phrases, *P#4* is not applicable to the following sequence:



Thus, a methodology to detect different manifestations of a phrase type is crucial for proper grammar induction and facilitates induction of a compact grammar (see Section 8.5.3).

8.5.2 Detection of Endocentric Constructions

A construction that is syntactically identical to its head is called *endocentric*. Those constructions contain exactly one head or several ones in symmetric coordinate constructions. Additionally to the head, at least one optional subordinating element has to be contained in the construction.

On the other hand, exocentric constructions do not contain a head element showing syntactic similarity to the whole construction. A classic example of an exocentric construction is a sentence when traditionally divided into subject noun phrase and predicate verb phrase (see Chomsky, 1957). Since a sentence is neither like a NP, nor like a VP, it is exocentric.

The distinction of these two mutually exclusive construction types reaches back to Bloomfield (1933). As all constructions in dependency structures are

necessarily endocentric, this distinction exists only in constituency grammars and therefore in PSG (see Osborne et al., 2011).

Following the definition of endocentricity, a phrase containing a head and an optional element should be equally distributed – in respect to its context – as the head. Consequentially, a phrase is considered to be endocentric, if it contains an element showing high context similarity (see Equ. 8.20).

$$\text{endocentric}(P) \Leftrightarrow \exists i : \text{sim}(\text{context}(P), \text{context}(p_i)) \geqslant \vartheta \tag{8.20}$$

The global context $\text{context}(P)$ of a phrase or POS tag P is the sum of all local contexts of P within the training corpus. For this task, local contexts are constructed out of two preceding POS tags (left neighbors) and the two following tags (right neighbors) including aforementioned markers for the beginning and the end of a sentence for each occurrence of P. The Cosine Measure is applied to calculate the similarity between the two contexts. If the resulting similarity passes a defined threshold ϑ, the phrase is considered to be endocentric.

Throughout all experiments, ϑ is set to 0.9 by heuristic. Some examples of English endocentric constructions are given in Table 13:

head		body
NNS	←	JJ NNS
NN	←	JJ NN
NNP	←	NNP CC NNP
NN	←	NN CC NN
VBZ	←	RB VBZ

Table 13.: Examples of endocentric constructions

8.5.2.1 *Impact on grammar induction*

Being able to detect heads along with their modifiers without any a priori knowledge facilitates unsupervised grammar induction in multiple ways.

The most obvious benefit is the reduction of complex structures. Without distinction between endocentric and exocentric constructions, at least two rules (P#1 ← JJ NN and P#2 ← JJ P#1) are necessary to parse the phrase *first civil settlement* as given in Figure 27a. Regarding this example, an additional rule for each additional adjective in front of the noun is necessary to parse the resulting phrase properly. This phrase is not complex in a linguistic perspective, but from a corpus linguistic point of view, its complexity grows gradually with the number of modifiers. While adjectives in front of nouns are very common and one of the most significant co-occurrences, multiple adjectives in front of nouns show another behavior. Although they are still common as it can be seen in Table 14[4], the significances of the co-occurrences of *JJ* and *JJ NN* are not very high and decrease with the growing number of adjectives. Thus, it is very important to detect those constructions despite the bottle neck provided by the data.

Number of adjectives	Frequency English	Frequency German
1	55518	48723
2	5261	3527
3	303	148
4	6	5
5		1

Table 14.: Statistics of adjective use in front of nouns

Using knowledge about subordinating elements leads to induction of a single rule (NN ← JJ NN) achieving the same parsing result (see Figure 27b). Additionally, the induced grammar circumvents some data-sparseness problems as no rare occurrences like JJ . . . JJ NN need to be contained in the training corpus to eventually parse those phrases.

4 The statistics are calculated on the Wall Street Journal Corpus and the TiGer Corpus for English and German, respectively. For English, tags denoting adjectives are *JJ* and *JJS*, tags denoting nouns are *NN*, *NNS*, *NNP* and *NNPS*. For German, tags denoting adjectives are *ADJA* and *ADJD*, tags denoting nouns are *NN* and *NE*. See Section A for further details about the tagsets.

This ability fulfills the *compactness* criterion of Minimum Description Length (MDL) (see Rissanen, 1978) for grammar generation providing a compact representation of the induced grammar and data. Linguistic argumentation favors short analyses (see Chang, 2004; Klein & Manning, 2002b) and minimal grammars due to analytical (see Harris, 1951) and cognitive (see Chomsky & Halle, 1968) economy.

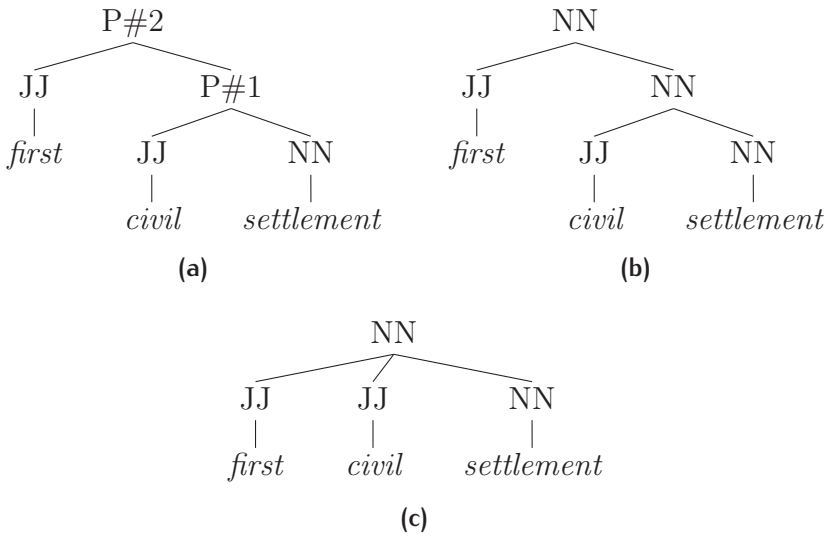


Figure 27.: Different structures for *first civil settlement*

Another aspect of using the induced knowledge about heads and modifiers lies in further processing steps. Besides the boost for grammar induction, the knowledge of which tags contain heads, which contain modifiers is very useful in e. g. RE (see Section 3.4). Removing optional modifiers from the sentence reduce complexity for subsequent analysis tasks.

This observation can be expressed in a generalized way: a transformation function $\Theta(S, G, K)$ can be applied to a sentence S, a grammar G and the derived knowledge K leading to various representations of a sentence. Different Θ-functions can be employed in certain tasks or scenarios. Those tasks include among other tasks:

RELATION EXTRACTION The task of Relation Extraction (see Section 3.4) does not need any knowledge about modifiers at all. Thus, simplifying the structure

of a sentence by removing dispensable words will improve the extraction accuracy as this transformation reduces noise in the data.

TRANSFORMATION BETWEEN DIFFERENT GRAMMAR THEORIES In order to compare structures that have been induced by parsers following different grammar theories, a transformation is essential. Particularly when evaluating the results of a parser to an annotated corpus, results can only be considered to be reliable if both structures follow the same grammar theory or annotation guidelines (e. g. the inner structure of ⟨JJ JJ NN⟩).

8.5.3 Phrase Type Clustering

One disadvantage of unsupervised parsing is the fact that different constitutions of phrases are not labeled the same way and thus, are not treated equally during the parsing step. An example is given in Figure 26a, where P#2 and P#3 obviously belong to the same phrase type (NP in this example). Of course, without a priori knowledge about language, it is not possible for the grammar induction algorithm to label phrases *NP*, *PP* or like any other known phrase types. Nevertheless, it is possible to cluster phrases into different phrase types to detect identical ones.

The linguistic argument for the existence of phrase types is *substitutability*. Phrases of the same type occur within similar contexts and can be mutually exchanged (see Harris, 1954; Radford, 1988).

For instance, *DT NN* and *DT NNS* given in Figure 26a occur in very similar contexts, and are both common noun phrases. Thus, the task of identifying phrase types is very similar to clustering words into classes for POS tagging purposes. Analogously to POS tag induction, syntactic context is used in any form of distributional clustering to define a similarity measure between two syntactic units (Schütze, 1995; Klein & Manning, 2002a). Given a proper measure on the basis of left and right contexts, distinction between several phrase types

is a much easier task than identification of phrases in the first place (Klein & Manning, 2002a; Haghighi & Klein, 2006).

Reichart & Rappoport (2008) propose a two-step approach. First, they apply the CCL-based algorithm Seginer (2007) to induce unlabeled bracketings. In a second step, they use the Bayesian Model Merging of Borensztajn & Zuidema (2007) to induce a Context-free Grammar (CFG) including labeling. Features presented in Stolcke & Omohundro (1994) and Petasis et al. (2004) where combined in this model.

As discussed in Section 8.3.1, a greedy algorithm for grammar induction is presented in this work. Consequentially, phrase type detection should be integrable into the induction process itself and should not be a separate post structure induction step as in most related work (Borensztajn & Zuidema, 2007; Reichart & Rappoport, 2008).

The presented algorithm for phrase type detection decides for each induced phrase whether it is an unseen phrase type at this point during the induction process or the constituent belongs to an already known phrase type reducing the problem to simply binary classification task.

A distributional similarity function is analogously defined as for detection of endocentric constructions (see Section 8.5.2) measuring the similarity between two global contexts:

$$\text{(8.21)}\quad sim_{distr}(P, Q) = sim(context(P), context(q))$$

If the similarity of phrase P (the one being tested) and Q (see most similar one, see Equ. 8.22) exceeds a threshold ϑ, then phrase P is considered to have the same phrase type as Q (see Equ. 8.23). In this case, P will be labeled by the label of Q and thus, is treated like Q in all following induction steps.

$$\text{(8.22)}\quad Q = \underset{q \in phrases}{arg\,max}\; sim(context(P), context(q))$$

$$\text{(8.23)}\quad Type(P) = Type(Q) \Leftrightarrow sim(P, Q) \geqslant \vartheta$$

8.5.3.1 *Do covered POS sequences matter?*

Besides distributional context the constituents themselves contain valuable information about their types – the POS tag sequence of the covered words. Regarding common noun phrases in the WSJ Corpus, approximately 79% contain at least one noun. Regarding prepositional phrases, even 98.5% contain a word tagged as preposition. [5]

Following this observation, parts-of-speech are obviously a determining factor for the superordinating constituent's phrase type. Furthermore, many linguistic and psycho-linguistic theories suppose some labeled hierarchical constituent structure arguing that there is observable cognitive-psychological evidence (see Goldberg, 2006).

$$\text{(8.24)}\quad sim_{POS}(P,Q) = \frac{\sum_{p\in P}\sum_{q\in Q} equ(p,q)}{|P||Q|}$$

$$\text{(8.25)}\quad equ_{exact}(p,q) = \begin{cases} 1 & p = q \\ 0 & else \end{cases}$$

$$\text{(8.26)}\quad equ_{fuzzy}(p,q) = sim(context(p), context(q))$$

Thus, two extended similarity functions are defined to incorporate contained parts-of-speech. $sim_{POSexact}$ compares the POS tags directly with each other, while $sim_{POSfuzzy}$ compares them measuring their respective context similarity. This is necessary to deal with fine-grained unsupervisedly induced POS clusters.

5 The WSJ Corpus contains 432776 noun phrases (label *NP**), 342319 of them contain at least one normal noun (tag *NN**) or a proper noun (tag *NNP**). The rest mostly consists of empty elements or personal pronouns. 117822 prepositional phrases (label *PP**) are contained in the WSJ, 116111 containing at least one preposition (tags *IN* and *TO*).

8.5.3.2 *Evaluation of Phrase Type Clustering*

In order to evaluate the proposed phrase type clustering similarity measures, an evaluation setup similar to the one for POS tagging (see Section 7) is applied.

All phrase prototypes[6] reaching a minimum frequency of 100 are extracted from a gold standard corpus (the top 10 for English and German are given in Table 15). The induced phrase types will be compared against the instances of these prototypes.

English		German	
prototype	frequency	prototype	frequency
DT NN	36080	ART NN	7649
NNP NNP	14543	APPR ART NN	3022
DT JJ NN	12041	ART ADJA NN	2359
JJ NNS	9716	NE NE	1888
IN DT NN	6619	APPRART NN	1774
DT NNS	6380	ADJA NN	1672
JJ NN	6127	APPR NN	1594
IN NN	5772	APPR ART ADJA NN	1037
NN NNS	5268	APPR NE	920
CD NN	5236	NN KON NN	829

Table 15.: The 10 most frequent phrase prototypes for English and German

For phrase type induction the frequencies of the POS sequences occurring within the aforementioned phrase instances are calculated and ranked in descending order. Note that for each language two evaluation runs are performed: one with the manually annotated POS tags of the gold corpus and one with tags induced by the unsupervised POS tagger described in Section 7.2. In the first case, the ranked POS sequences are identical to the most frequent phrase proto-

6 POS sequences of constituents

types. In the latter case, much more POS sequences will be necessary to cover all phrase instances of the evaluation set.

For each pattern (in descending order), a phrase type is determined. If the pattern shows high similarity to an already induced phrase type, this type is attached. Otherwise, a new phrase type will be used to label this phrase prototype.

The accuracy is calculated denoting the portion of correctly merged phrase prototypes into a phrase type cluster. Each proposed similarity measure for phrase type clustering is evaluated separately. sim_{distr} is the basic similarity measures, the other two extensions function as a filter: if sim_{POS} is smaller than 0.5, the similarity score is set to zero.

	sim_{distr}	$+sim_{POSexact}$	$+sim_{POSfuzzy}$
English	0.76	0.81	0.77
German	0.89	0.92	0.90

(a) Gold Corpus Tags

	sim_{distr}	$+sim_{POSexact}$	$+sim_{POSfuzzy}$
English	0.82	0.86	0.83
German	0.93	0.98	0.95

(b) Unsupervised Tags

Table 16.: Phrase type clustering evaluation results for English

The accuracy is improved in all cases. Exact POS tag matching yields the highest accuracy gain. As for most tasks, this increase results in decreasing the number of correctly extracted prototype pairs. As this decrease is of much higher magnitude ($> 50\%$) than the accuracy gain, contained POS tags are not incorporated into the phrase labeling process.

8.6 ITERATIVE GREEDY LEARNING

The grammar induction process is realized as an iterative greedy algorithm. A pseudo code description of the proposed process is given:

```
data = tagged corpus
init rules
while a new rule exists
  nPhrase = find valid phrase candidate
  nLabel = find label for nPhrase
  nRule = create new rule from nLabel and nPhrase
  add nRule to rules
  apply rules to data
```

First, the grammar contains no rules. With each iteration, at least one rule will be added to the grammar until the grammar induction process is complete. As long as new rules exist, the best constituent / phrase candidate (*nPhrase*) is detected. Afterwards, the proper label is defined. The current phrase candidate could be endocentric, belong to an existing phrase type or could be the first observed prototype of a new phrase type. After all necessary information for the next grammar rule has been collected, the new rule will be added to the rule set of the parser model.

During finalization of each iteration, the data will be parsed using the so far induced grammar. This step reduces the complexity of the data and allows detection of complex structures with tree depth of two or more. The parsed corpus is the input of the next iteration.

The complete induction process will be demonstrated using the example sentence *The minimum unit is $ 100.*

The initial structure is created by addition of the sentence's words as leaves under a common root node. During the first iteration, the best grammar rule is added to the parser model.

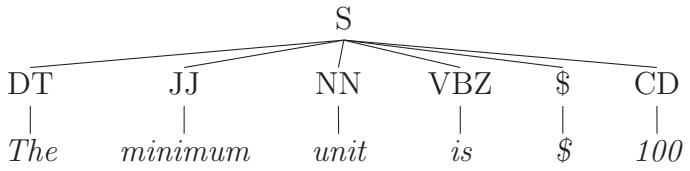


Figure 28.: Initial structure

Finalizing the first iteration, the newly induced rule is applied and syntactic structures emerge.

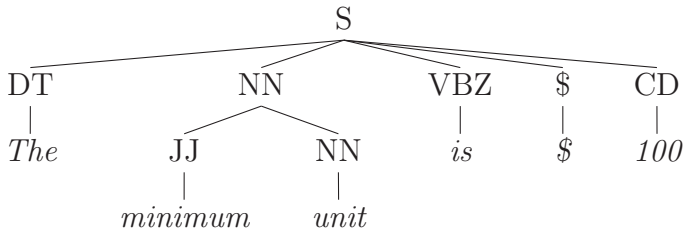


Figure 29.: Resulting structure after application of rule *NN ← JJ NN*

The induction algorithm uses the current structures during the next iteration for co-occurrence calculations and phrase candidate extraction. During each step, the top-most level under the root node of all sentence structures is used. The POS sequence of this example is *DT NN VBZ $ CD*.

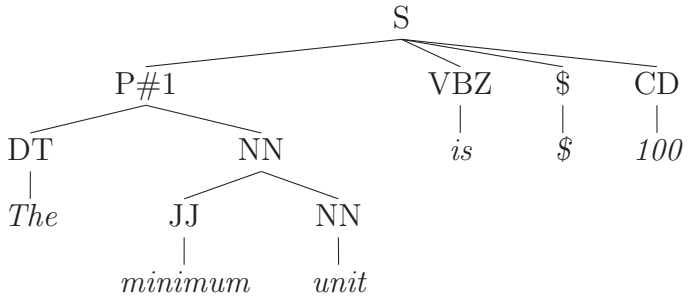


Figure 30.: Structure after application of rule *P#1 ← DT NN*

After two further rules, the final structure of the sentence has been induced (see Fig. 32).

The learning process will be aborted if there are no valid rules left that could be induced. The process is considered to be completed if one of the following three conditions is met:

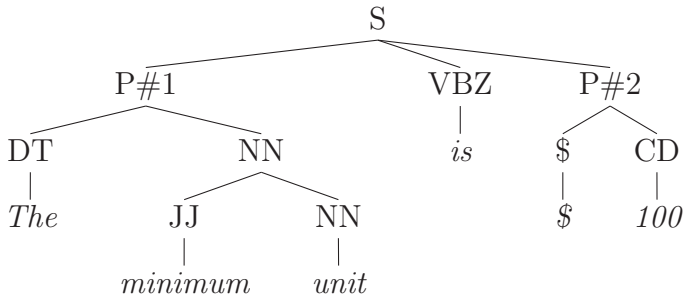


Figure 31.: Structure after application of rule *P#2 ← $ CD*

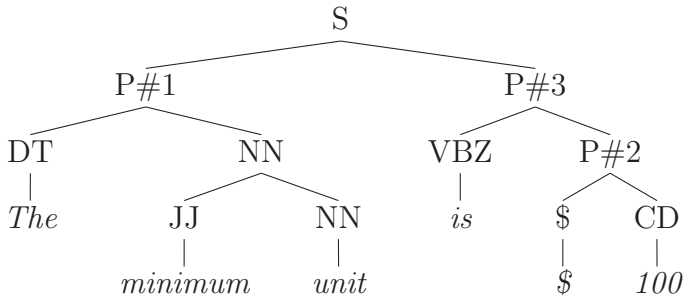


Figure 32.: Structure after application of rule *P#3 ← VBZ P#2*

THE MAXIMUM NUMBER OF RULES HAS BEEN MET The algorithm induced the previously defined maximum number of rules.

NO SIGNIFICANT RULES ARE LEFT No phrase candidate is considered to be significant and thus, no further rules can be induced.

THE CORPUS HAS BEEN PARSED COMPLETELY All sentences of the training corpus are reduced to one phrase and thus, the corpus has been parsed completely.

The induced context-free or context-sensitive grammar (depending on the applied function for candidate scoring (see Section 8.4.2) is given as a list of grammar rules. Two induced example grammars for English and German are given in the Appendix (see Sections B.1 and B.2 for English and German grammars, respectively).

The according parsing algorithm applies the grammar rules in order of significance to the sentence and shapes structures. As the rules are induced in descending order of significance, the parsing of sentences is very similar to the induction process.

8.7 PARSER EVALUATION

The evaluation setup is the one presented in Sec. 8.2. Results are presented for grammar induction on word strings (*unsuParse (words)*) and on manually annotated POS tags provided in the evaluation data (*unsuParse (POS tags)*).

Comprehensive results are given for German data as the major application is parsing the German AIM Corpus for RE. Scores for further approaches to unsupervised parsing are presented for comparison:

Parsing Model	UP	UR	UF
CCM	48.1	85.5	61.6
DMV + CCM	49.6	89.7	63.9
U-DOP	51.2	90.5	65.4
UML-DOP	—	—	67.0
U-DOP*	—	—	63.8
unsuParse (POS tags)	**71.1**	**67.9**	**69.5**
Common Cover Links	51.0	69.8	59.0
unsuParse (words)	**63.1**	**60.4**	**61.7**

Table 17.: UP, UR and UF for NEGRA10

unsuParse achieves the best results when being compared to the other approaches. It achieves the highest precision among all models. As most other models (except for CCL) induce binary trees, this is not very surprising. Consequently, recall is lower than achieved by binary trees. Regarding f-score, *unsuParse* obtains the highest f-score for German data.

When being applied to more complex sentences (NEGRA_{40}), the performance drops. Both precision (55.3%) and recall (51.4%) score much lower than on shorter sentences.

Constituent-Chunk scores achieve similar results as presented by Ponvert et al. (2011). Both approaches make similar assumptions and thus, differ only slighty. *unsuParse* achieves a CC precision of 41.4% and a CC recall of 51.2% which is marginal better than presented in Ponvert et al. (2011). The difference on the chunk level originates in the more restrictive constituent scores of *unsuParse*. The same reason explains the lower recall regarding deeper structures on which *unsuParse* yields lower recall than other approaches.

Table 18 presents the most frequently over- and under-proposed phrases induced on NEGRA10. Especially *NPs* and *PPs* are often over-proposed due to the flat structures provided by the NEGRA corpus (see Sec. 5). The highest ranked under-proposed phrase *NE NE* is correctly learned as phrase and classified as endocentric construction (NE ← NE NE). Due to removed punctuation which normally separates enumerations, separated proper nouns be represented by one flat phrase. Such effects would not occur when parsing natural language containing punctuation.

Overproposed		Underproposed	
ART NN	369	NE NE	42
CARD NN	111	NN NE	35
ADV ADV	103	ART NN NE	27
ADJA NN	99	ADV ART NN	24
APPR ART NN	93	APPR PPER	23

Table 18.: Most frequently over- and under-proposed constituents

Another frequent error is the induction of verb phrases such as ⟨MD VB⟩. This co-occurrence is very significant and thus, they are considered to build a phrase. This type of error states a general problem and is also reported by Klein & Manning (2002a).

8.8 PREDICATE DETECTION

Investigating syntax of a language includes more than the induction of part-of-speech tags and the according grammar. Using unsupervised methods leads to difficult challenges at some points. Although, several tasks of unsupervised learning of syntactic features of a language have been successfully dealt with, many challenges remain unmastered. One of those challenges is the unsupervised predicate argument detection.

The task of predicate argument detection can be divided in several subtasks.

PREDICATE DETECTION Verbs are the predicates / relations in natural language. According to this definition, knowing the verbs of a language is essential for predicate detection. Using supervised learning methods based on manually annotated corpora bypasses this subtask as verbs can be detected by extraction of all words tagged with an appropriate POS tag. Even information about the number of θ-roles can be learned from language resources for certain languages, such as the *PropBank* for English (see Palmer et al., 2005).

ARGUMENT DETECTION Different syntactic entities can occupy a θ-role of a predicate. Regarding relation extraction as the next step to unsupervised knowledge extraction, nominal constructions are the ones to focus on. This includes noun phrases of all kinds, such as normal noun phrases, named entities, pronouns etc. Also prepositional phrases are of interest due to the embedded nominal expression and the additional information about possible role types provided by the preposition itself.

PREDICATE–ARGUMENT DETECTION The third subchallenge deals with the actual problem. Knowing the predicates and the possible arguments solely needs to be completed by the assignment of arguments to their corresponding predicate. Probabilistic models about the number or types of θ-roles which have to be assigned by a predicate can achieve several improvements.

Besides sophisticated relation extraction models, lexicalized grammar induction can be improved using statistical models of verb argument structures.

This section deals with the first task – verb detection in a completely unsupervised manner.

As stated before, verbs represent natural language relations between various arguments. These arguments primarily are nominal or prepositional phrases and nominal or prepositional expressions, respectively. Such phrases characteristically show a shallow inner structure and their words are bond through high-significant neighborhood co-occurrences. Consequentially, approaches to chunking (see Skut & Brants, 1998) or shallow parsing (see Hänig et al., 2008; Hänig, 2010a) are able to extract the aforementioned phrase types.

Since completely unsupervised approaches do not incorporate any a priori knowledge about the language in question, it is not clear, which phrases contain arguments for verbs. Not even if the unsupervised parser induces labeled parse trees as in Reichart & Rappoport (2008).

The major assumption behind this approach to unsupervised verb detection is that each sentence contains at least one subject and at least one predicate. This relation between subjects and predicates originates in Latin and Greek grammars and is based on term logic[7].

8.8.1 Tag Sequence Alignment

The Tag Sequence Alignment (TSA) algorithm (see Hänig, 2011) employs *unsuParse* to detect arguments. A very strict configuration for unsuParse is used to stop grammar induction right after all phrases yielding a very high constituent score have been induced.

7 Term logic is also known as Aristotelian logic and leads back to Aristotle.

This leads to a crucial reduction of the sentences' complexity. The example *The man lost his purse during the game.* will demonstrate this reduction[8]. Its proper syntactic tree is given in Fig. 33.

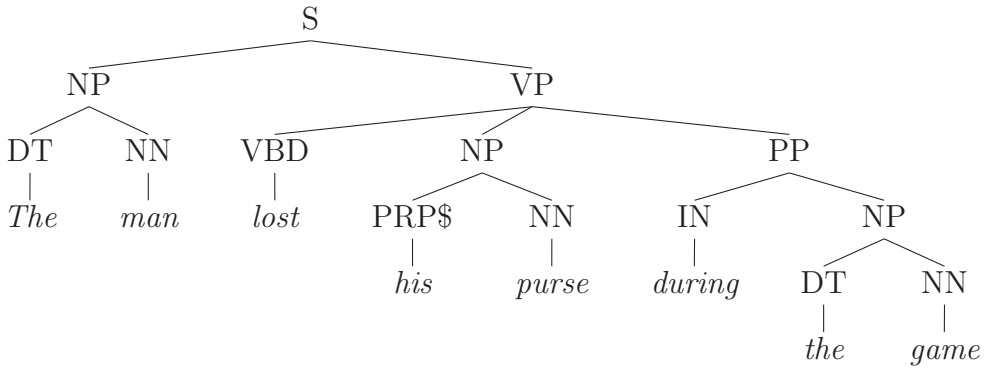


Figure 33.: Syntactic tree for *The man lost his purse during the game.*

Employing the shallow parser to reduce the complexity of this sentence leads to the following analysis (see Fig. 34).

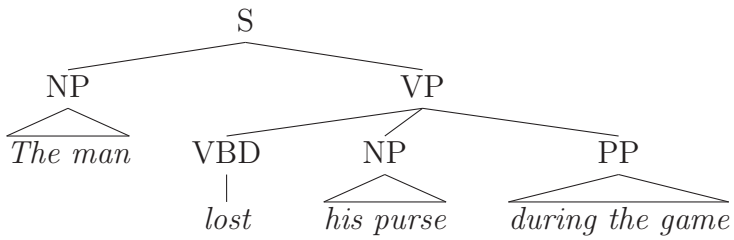


Figure 34.: Reduced syntactic tree for *The man lost his purse during the game.*

8 The punctuation mark is omitted due to focusing on the words.

Assuming that verbs dominate the structure of sentences decisively as proposed in dependency grammars, the arguments can be exchanged, moved to different positions and even be omitted:

The man	lost	his purse	during the game.
The woman	lost	her hat	in the park.
The student	bought	a notebook.	

Figure 35.: Examples for exchangeability, movement and omission of arguments.

Each of the three sentences is grammatically sound while they show high similarity in syntactic structure. TSA exploits this property – which is very distinct for sets of short sentences – and aligns simplified tag sequences to each other.

In a formal way, a sequence s of a sentence with length n is defined as:

$$s = (s_0 \ \dots \ s_{n-1}) \tag{8.27}$$

An element s_i of such a sequence can be a either a phrase tag or a POS tag. The corresponding sequence s for the example given in Fig. 34 is a concatenation of the pre-terminal nodes of the syntactic tree, in particular: $s = (\text{NP VBD NP PP})$.

In order to get to a superordinate syntactic level, tags can be grouped. These groups represent phrases. Each phrase of a sentence s can contain any number of tags – within the boundaries defined by s – and thus, 2^{n-1} possible groupings exist for s. Between the two extremes, a grouping containing exactly one group (e. g. $((\text{NP VBD NP PP}))$) and a grouping containing n groups (e. g. $((\text{NP}) (\text{VBD}) (\text{NP}) (\text{PP}))$), several groupings can be created (e. g. $(((\text{NP}) (\text{VBD}) (\text{NP PP})))$.

Formally, each grouping of a sentence can be defined by a sorted set of separation indices I contained in the power set PI:

$$PI(n) = P(\{0 \ \dots \ n-2\}) \tag{8.28}$$

Each possible grouping $g(s, I)$ of a sentence with length n is defined as given in Equ. 8.29:

$$G(s, I) = \left((s_0 \dots s_{i_0}) (s_{i_0+1} \dots s_{i_1}) \dots \left(s_{i_{|I|-1}+1} \dots s_{n-1} \right) \right) \tag{8.29}$$

Because each sentence contains at least one predicate, at least one group of each possible grouping G for sentence s has to represent a predicate. This predicate is detected by finding the best alignment of two different sentences s and t – assuming that sentences show similar structure with regard to exchangeability, movement and omission of arguments (see Fig. 35).

The similarity of two groupings is defined accordingly:

$$\mathrm{sim}_{seq}\left(G\left(s,I\right),G\left(t,J\right)\right)=$$

$$\begin{cases} |I| \neq |J| : & 0 \\ \nexists k : G\left(s,I\right)_k = G\left(t,J\right)_k \wedge type\left(G\left(s,I\right)_k\right) = \mathrm{POS} : & 0 \\ else : & \\ \frac{1}{|I|}\sum_{i=0}^{|I|-1} \mathrm{sim}\left(G\left(s,I\right)_i, G\left(t,J\right)_i\right) & \end{cases} \tag{8.30}$$

The first case of $\mathrm{sim}_{seq}\left(G\left(s,I\right),G\left(t,J\right)\right)$ deals with the number of phrases per sentence. The number of phrases in s and t has to be equal to hold the condition $\forall g_i \in G\left(s,I\right) : 0 \leqslant i < |G\left(t,J\right)|$ which is essential for proper alignment computation. If this condition is not met, a proper alignment cannot be ensured. Thus, the similarity score of these alignments is 0.

The predicate can either be a single word (e. g. *talk*) or a compound (e. g. *was talking*). As compound predicates consist of at least one simple verb form, TSA will focus on detection of single-word predicates and deal with the detection of multi-word predicates in a subsequent step. Case two of $\mathrm{sim}_{seq}\left(G\left(s,I\right),G\left(t,J\right)\right)$ makes sure that these requirements are met.

The last case calculates the actual similarity. It is defined as the average of the context similarities of all conjugated group pairs of the sentences' groupings. Context similarity is calculated using the cosine measure.

The best alignment for two sentences s and t is the pair of indices (i,j) : $\underset{(i,j)}{\mathrm{argmax}}\ \mathrm{sim}_{seq}\left(G\left(s,i\right),G\left(t,j\right)\right)$ s. th. $i,j \in PI\left(|G\left(s,i\right)|\right)$.

Within the resulting two groupings, the group fulfilling criterion two of Equ. 8.30 is considered to be the best predicate candidate.

8.8.2 Detection of verbs in a corpus

To eventual extract all parts-of-speech that tag predicates, a complete corpus needs to be processed. First, the corpus is tagged and parsed completely. Afterwards, each sentence of the corpus is transformed into its corresponding sequence. To avoid effects from rare sequences, only sentence sequences showing a minimum support $\eta \geqslant 10$ are taken into account. As the detection takes place at the POS level, data-sparseness problems which could occur at the word level can be neglected.

In descending order of frequency, all patterns occurring at least η times in the corpus will be aligned. The most frequent pattern is treated as seed pattern and thus, simply split into its components (e. g. ⟨NP⟩ ⟨VBD⟩ ⟨NP⟩).

Each further sequence will

(a) either be aligned to a very similar pattern, or

(b) treated as seed pattern of a new sentence pattern.

In scenario (a), the similarity sim_{seq} between the known sequence and the new one has to pass a threshold ϑ in order to be considered being similar. If this is the case, the most similar grouping G of the new sentence sequence will be associated with the corresponding seed sequence (e. g. ⟨NP⟩ ⟨VBD⟩ ⟨NP PP⟩). Scenario (b) equals the initialization at the beginning of the alignment. The new sequence is considered to belong to a new, not yet observed type of sequences and thus, is split into its components and creates a new possible alignment.

With each processed sentence, the resulting alignments are shaped and formed due to the language characteristics. Each alignment has a fixed point within its graph representation. According to the underlying hypothesis of TSA, this fixed point contains the predicate node (e. g. in Figure 36).

The part-of-speech in the center of the graph (as VBZ in the example) is considered to occupy a central role within the sequences. Thus, all parts-of-speech showing this hub property in any extracted alignment will be marked as verb.

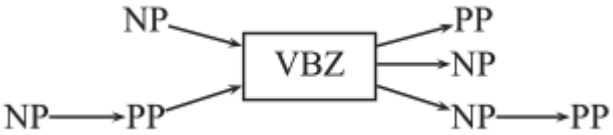


Figure 36.: Tag Sequence Alignment for three common sequences.

8.8.3 Tag List Expansion

Due to the restrictions made during the alignment process (e. g. minimum support η, minimum sequences per alignment), it is not possible to detect all tags containing verbs directly.

After the application of TSA, a set T_P is obtained containing all tags marking predicates. Using these tags, a set W_P can be created by putting all words of a corpus into it, that has been tagged at least once by any tag of T_P.

The fact, that many words occupy different syntactic roles within a sentence (see motivation for disambiguating POS tagging in Sec. 7), is used to detect further tags marking predicates.

Each part-of-speech epitomizes a syntactic function or role and thus, all words tagged by a certain part-of-speech have identical syntactic functions[9]. Using the inversion of this argument, each tag marking more predicates than other words is considered to mark predicates.

Correspondingly, the predicate coverage $cov(t)$ is calculated for each tag t (see Equ. 8.31).

$$cov(t) = \frac{|words\ annotated\ by\ t \cap words \in W_P|}{|words\ annotated\ by\ t|} \tag{8.31}$$

All tags achieving coverage of at least 50% – which is the natural choice – are considered to mark predicates, too. Eventually, tag set T_{Pe} denotes the extended tag set containing T_P and all tags t holding $cov(t) \geqslant 0.5$.

9 Assuming perfect part-of-speech tagging.

8.8.4 Evaluation

The evaluation of predicate detection is performed on two different setups: a supervised and an unsupervised one. Although TSA is a completely unsupervised approach, it is also evaluated on unsupervised tags to provide comprehensive results reducing the probability of tagging errors.

Evaluation is performed on two corpora, one for English and one for German. Both corpora were created by *Projekt Deutscher Wortschatz* (see Quasthoff et al. (2006)) and contain 1 million sentences each.

The supervised setup's purpose is to eliminate erroneous influences from the preprocessing steps and to provide high quality prerequisites.

SUPERVISED SETUP The supervised setup employs the Stanford POS Tagger (see Toutanova & Manning, 2000) and the Stanford Parser (see Klein & Manning, 2003b). Sentence patterns are created by extraction of all kinds of prepositional phrases and noun phrases at the topmost level.

UNSUPERVISED SETUP In the unsupervised setup, unsupervised approaches to POS tagging (see Section 7) and parsing (see Section 8) are applied. First, all words of the corpus are tagged and afterwards, a model for unsupervised parsing is trained using a strict significance threshold of $\eta = 0.3$. Analogously to the supervised setup, sentence sequences are extracted from the induced parse trees using the labels from the topmost nodes beneath the root nodes.

In either configuration, all sentence sequences occurring at least 10% as often as the most frequent one, are taken into account. The similarity threshold ϑ was set to 0.8.

8.8.4.1 *Part-of-speech tagsets*

The evaluation can be performed straight forward. Each part-of-speech tag t out of the extracted set T_P or T_{Pe} either marks verbs or not. In order to be able to

decide, each of used tag sets will be introduced shortly. Basicly, a distinction between supervised and unsupervised annotation with POS tags can be made.

SUPERVISED ANNOTATION Supervised POS tagger are trained on manually or semi-supervised annotated data and thus, replicate the tags observed in the training data. For most languages, for which language resources exist, an established tagset is used. For the present evaluation, two tagsets are of special interest: the Penn Tree Tagset (see Santorini (1990)) for English and the Stuttgart-Tübingen Tagset (see Thielen et al. (1999)) for German respectively.

Table 19 gives a brief overview over the tags marking verbs[10]. Additionally to the 7 English verb tags and the 12 German verb tags, their relative frequency (compared to all verbs in the respective corpora) is given. These frequencies are used to calculate the coverage of the extracted tag sets T_P and T_{Pe}.

UNSUPERVISED ANNOTATION Unsupervised part-of-speech taggers do not label word classes in a classical, human-readable way. Word clusters are enumerated instead. Hence, the short overview about word clusters containing verbs contains additionally to cluster number and relative frequency some members of this class to describe the cluster appropriately (see Table 20 for English and Table 21 for German). As there are numerous word clusters, only the most frequent ones are shown at this point.

8.8.4.2 *Results*

We calculated precision and recall scores for the extracted verb classes (see Tables 22 and 23), the corresponding tag sets are given in Tables 24 and 25.

For both the supervised and unsupervised data sets a precision of 100% is achieved as all extracted tags denote verbs. Regarding supervised data sets for English and German, TSA extracts 55.3% and 78.9% of all verbs. Expanding the extracted tag sets yields a significant improvement for English (increasing recall

10 A comprehensive overview over all tags of both tagsets is given in Appendix A.

Penn Tree Tagset		STTS	
Tag	Rel. Frequency	Tag	Rel. Frequency
MD	6.05%	VAFIN	24.74%
VB	18.21%	VAIMP	0.00%
VBD	26.81%	VAINF	2.67%
VBG	10.51%	VAPP	1.17%
VBN	15.99%	VMFIN	7.81%
VBP	9.48%	VMINF	0.18%
VBZ	12.95%	VMPP	0.01%
		VVFIN	34.04%
		VVIMP	0.06%
		VVINF	12.27%
		VVIZU	0.98%
		VVPP	16.07%

Table 19.: Established verb tags for English and German

to 89.4%), while the benefit on German data is marginal. This effect originates in the morphological richness of German.

The results on unsupervised data achieved perfect precision, too. Tag list expansion does not have a measurable impact on these results (about 0.02%) and can be neglected. However, expansion adds some classes including some incorrect ones[11]. The inferior recall results from a larger number of different word classes induced by unsupervised POS taggers.

11 Incorrect classes are presented by italic font in Table 24 and Table 25

Tag	Description	Relative frequency
6	classify, let, sustain	20.82%
15	navigating, expending	8.75%
18	elaborates, transports	6.29%
26	underlined, subdivided	34.85%
185	may, will, might	5.27%
342	can, could, should	2.66%
445	be	5.27%
478	are	2.90%
479	is	6.26%

Table 20.: unsuPOS classes for English verbs

8.9 UNSUPERVISED PARSING – CONCLUSIONS

In this section, an approach to unsupervised parsing is presented. Evaluation proves that it achieves competitive results.

Although no a priori knowledge describing the language in question is taken into account, this approach generates parses and fulfills the previously desired features. It detects heads and modifiers and induces intelligent phrase labels. Different prototypes of the same phrase type are labeled by the same label and thus, the information about syntactic similarity is retained for further processing. Due to its greedy approach relying on simple syntactic rules, it is possible to easily alter induced grammars without huge effort. This enables semi-supervised grammar induction for new languages or data sources ensuring high quality analyses.

To facilitate tasks like RE, verbs are automatically detected which is a huge step towards fully unsupervised predicate-argument detection.

There is still potential space for improvements. First, deeper investigations into lexicalization can improve handling of exceptions in the sense of atypical

Tag	Description	Relative frequency
9	fragten, beteten	7.88%
10	spürte, ahnte, haste	5.77%
22	distanzierte, vertritt	3.88%
37	erfüllt, verringert,	16.03%
42	zugucken, dauern,	28.37%
200	hat, hatte, will, macht	4.98%
253	wird, wurde, erscheint	3.24%
334	ist, war, wäre	7.43%
380	sind, waren, seien	2.97%

Table 21.: unsuPOS classes for German verbs

Setup	Precision	Recall	F-Measure
supervised TSA	1.000	0.553	0.712
supervised TSA + TLE	1.000	0.894	0.944
unsupervised TSA	1.000	0.440	0.611
unsupervised TSA + TLE	-	-	-

Table 22.: Predicate detection results for English

Setup	Precision	Recall	F-Measure
supervised TSA	1.000	0.789	0.882
supervised TSA + TLE	1.000	0.816	0.899
unsupervised TSA	1.000	0.627	0.771
unsupervised TSA + TLE	-	-	-

Table 23.: Predicate detection results for German

Supervision	TSA	TLE
supervised	VBD VBP VBZ MD	VBN VB
unsupervised	26 478 479	112 126 *336* 350

Table 24.: Extracted POS tags for English

Supervision	TSA	TLE
supervised	VVFIN VVINF VAFIN VMFIN	VAINF VVIMP
unsupervised	9 37 42 334 380	135 *142 166 175* 230 ...

Table 25.: Extracted POS tags for German

structures. In order to induce deeper structures, a hybrid approach seems to be the best way. A combined model based on *unsuParse* and CCM could benefit from both advantages: high precision and deep structures obtained through induction of binary parses.

Regarding unsupervised verb detection, incorporation of dependency parses could improve the overall performance as DGs are based on a similar assumption attesting verbs a special function within sentences. Additionally, morphological analysis can facilitate the extraction through TSA on morphologically rich languages.

9

SYNTACTIC RELATION EXTRACTION

Syntactic relation extraction solely relies on syntactic annotations like POS tags and parsetrees which can be annotated in an unsupervised manner. Contrary to most other approaches to relation extraction, the syntactic approach presented in this thesis does not need any training samples. Instead, it exploits language statistics about structure.

An approach for syntactic RE has been presented previously by Hänig & Schierle (2009) with partially the same objective – extract automotive relations from repair order data. This originally approach achieved remarkable results. Nevertheless, two major advancements of this approach will be presented in this section.

First, the dependency to manually created *relation definitions* is removed. Those predefined heuristic rules define:

- two *category types* of the involved entities,
- a *preferred direction* to resolve ties between two relation candidates exploiting the linear order of the entities within the sentence,
- the *maximum distance* for this relation, and
- a flag marking whether this relation may be part of a complex relation (beyond binary relations).

In order to complete this task, statistics about entity distributions are thoroughly studied.

The second advancement is the application of syntactic RE to data gathered in internet fora discussing automotive topics. Beneath significantly different language properties, an additional entity type will be explored: polar utterances. The employed sentiment analysis extracts polar utterances on the phrase level resulting in manifold entities regarding to their syntactic properties (e. g. different phrase types) and thus, heterogeneous distribution within the data.

9.1 SEMANTIC TAGGING

As described in Section 6 parsing is a difficult task for this kind of data. Furthermore, as many terms consist of more than one token, parsing splits entities making it infeasible to determine the correct position of an affected entity within a parse tree.

Hänig & Schierle (2009) propose a combination of syntactic and semantic tagging to circumvent this problem. Before training a POS tagger, all relevant terms are replaced by their respective category label.

Basically, two classes of entity types can be distinguished:

THESAURUS CONCEPTS Concepts contained in a domain-specific thesaurus (see Sect. 5.5) are replaced by the name of their category (e. g. *combustion engine* will be replaced by COMPONENT).

SPECIAL TOKENS Tokens containing numerical values (e. g. numbers, mileage values, monetary values), temporal expressions (e. g. time and data values) and mixed tokens (e. g. codes for spare parts and corrective actions) are replaced by an appropriate label.

Example: all concepts of the sentence

> customer states left marker lamp warning light comes on when driving

will be replaced and the sentence is converted into:

customer states LOCATION COMPONENT SYMPTOM CONDITION

An unsupervised POS tagger (see Sec. 7) is trained on the converted corpus. Words contained in a class induced by an unsupervised POS tagger occur in syntactically similar contexts and show semantically similar properties. This is a beneficial side effect as clusters that contain a certain concept category often contain words belonging to this category which are not covered by the thesaurus yet.

An example of replaced concepts and annotated word clusters[1] is given in Fig. 37:

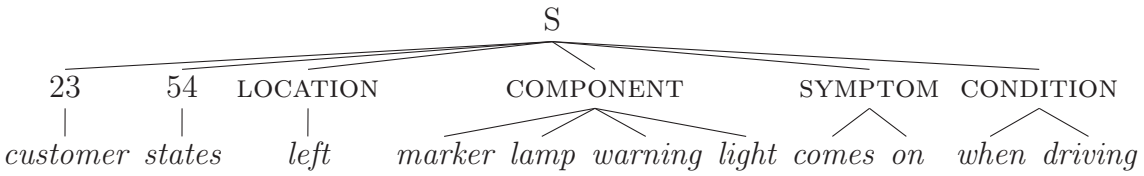


Figure 37.: Example for replaced automotive concepts

In order to induce parse trees for the corpus, all sentences are tagged and grammar induction as described in Sec. 8 is applied. A resulting parse tree is shown in Fig. 38:

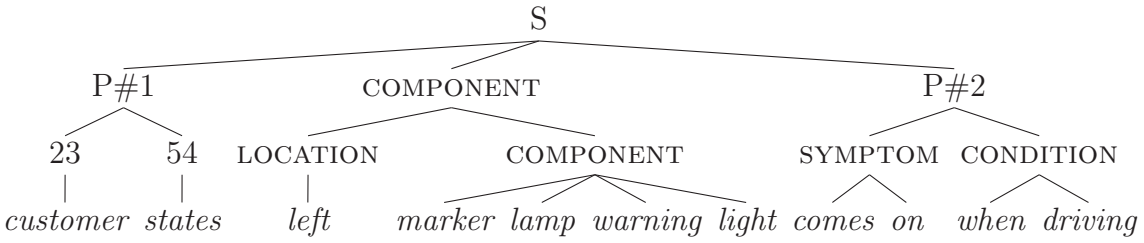


Figure 38.: Example for semantically tagged and parsed sentence

9.2 SYNTACTIC FEATURES

Given a sentence and its parse tree, two different syntactic measures are defined:

1 Word clusters containing a category are denoted by this category instead of the corresponding number for better readability.

TOKEN DISTANCE The token distance $d_{token}(e_1, e_2)$ between two entities e_1 and e_2 is defined as the number of tokens between those two entities. For this purpose, the entity layer provides an abstraction from the word level. Many entities consist of more than one word (e. g. COMPONENT *marker lamp warning light*) although they represent one semantic and syntactic unit. Thus, all entities are treated as a single token for calculation of d_{token}. Regarding the example of Fig. 38, d_{token}(LOCATION *left*, CONDITION *when driving*) is 2 as there are two *units* between these entities.

TREE DISTANCE The tree distance $d_{tree}(e_1, e_2)$ is defined as the length of the path between the two entities' nodes in the syntactic tree. The motivation for this measure originates in long sentences of the AIM Corpus. Entities often show huge token distances caused by subordinate clauses, enumerations or further reasons. The tree distance d_{token}(LOCATION *left*, CONDITION *when driving*) of the example given in Fig. 38 is 4 as four connections have to be passed.

9.3 LEXICALIZATION

Models for RE can be defined at two different ways regarding levels of language – in a *lexicalized* and *unlexicalized* fashion. Parsers are often distinguished between those two possibilities. The model of lexicalized parsers is defined on words themselves while unlexicalized ones employ an abstracting layer, such as POS tags. In terms of RE between entities is kind of unlexicalized per se as annotated entities abstract from words. Feature selection still uses words directly (see Rosenfeld & Feldman, 2006a).

In terms of RE between hierarchically structured entities, an additional possibility of abstraction exists. Theoretically, an abstraction can be made with each level of the entity hierarchy. In the following, two different abstraction levels

are studied: the concept level (the lowest one) and the category level (the most generic one).

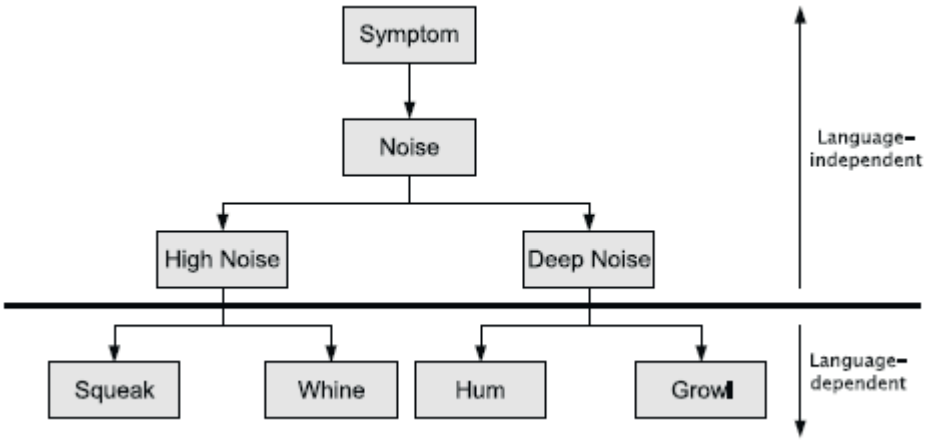


Figure 39.: Concept hierarchy: *noise*

The example *noise* given in Fig. 39 (see Schierle & Trabold, 2008) will demonstrate the two different levels of interest. The concept level describes the language dependent part of the hierarchy, meaning *Squeak*, *Whine*, *Hum* and *Growl*. Comparing concepts automatically deals with inflections and synonyms. The category level is the highest concept given in this example (*Symptom*). It can be used to generalize and aggregate statistics on a less detailed level.

9.4 DISTRIBUTION OF ENTITY PAIRS

In order to build a probabilistic model certain corpus statistics have to be calculated with respect to relations between entities. As no suitable annotated training corpus exists[2] choosing a strategy to pick training data states the first challenge.

Without any supervision throughout the process of training set generation, two heuristics can be employed.

2 The evaluation corpora contain appropriate annotations, but cannot be used for calculation of reliable statistics due to its limited size.

The intuitive one is to select all sentences S containing exactly two entities e_1 and e_2. Under the assumption that these two entities are related to each other (given the condition that no further automotive entity exists), each sentence $s \in S$ is considered to be a training instance for the relation between $type(e_1)$ and $type(e_2)$. $type(e)$ thereby denotes the concept or category of the instance, depending on the chosen granularity. Although it is very probably, no guarantee exists that both entities are related to each other. With growing token and tree distances, the probability for being related decreases. It would be possible to extract only those sentences containing exactly two entities occurring within a defined maximum distance, but this discards valid relation instances and thus, biases statistics towards narrow relations.

Another disadvantage of this heuristic is data sparseness. It can be observed for rare relations (such as SYMPTOM - CONDITION) which barely occur in the training set due to the restriction of sentences containing exactly two entities. Especially sentences contained in the QUIS Corpus barely contain only two entities. Calculation on concept level increases data sparseness problems.

The other heuristic is to use all sentences S containing at least two entities. The approach to take all possibilities into account when it is not clear which are the correct ones has proven to be successful for other NLP tasks (i. e. for parsing, see Bod, 2006a). The resulting training set $tset(S)$ is defined as:

$$tset(S) = \bigcup_{s \in S} \{(e_1, e_2) \in E_s \times E_s \wedge e_1 \neq e_2\} \tag{9.1}$$

where E_s denotes the entities contained in sentence s. This heuristic ensures that all possible relations are covered despite their arbitrariness. Furthermore, the available data is represented comprehensively and thus, probabilistic models defined on such a large data set yields superior results than on a sparse data set.

In this section, the distribution of entity pairs is explored unveiling differences between different relations and corpora. Based on these observations, a probabilistic model is motivated and defined so that the observed characteristics are covered by this model.

9.4.1 Preferred Order of Entity Pairs

As stated by Hänig & Schierle (2009), different automotive relations show variations regarding the preferred order of involved entities. This can be confirmed empirically. Fig. 40 shows the token distributions (calculated on the QUIS Corpus) of two relations COMPONENT - SYMPTOM and COMPONENT - LOCATION , respectively. The graph of the distribution for relation COMPONENT - SYMPTOM reveals the preference of the linear order COMPONENT followed by SYMPTOM [3]. The contrary observation can be made for the relation COMPONENT - LOCATION , in which LOCATION precedes COMPONENT in most cases.

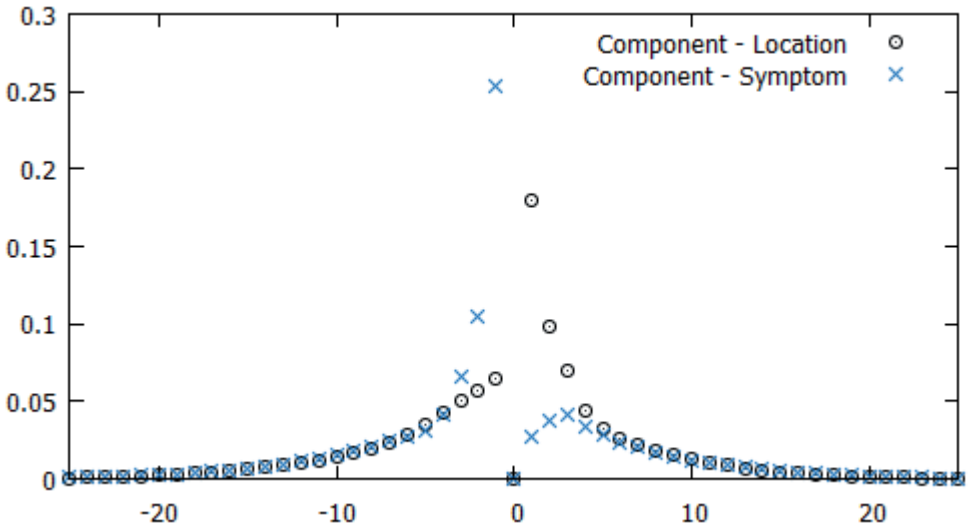


Figure 40.: Token Distribution of COMPONENT - SYMPTOM compared with COMPONENT - LOCATION

Statistics like this graph empirically validate the existence of preferred entity orders within automotive relations. Consequently, incorporation of a feature for linear order of the involved entities similar to a preferred direction is highly desirable for any model.

3 The x-axis of the graph shows the token distance of COMPONENT in relation to SYMPTOM and LOCATION , respectively. Negative numbers represent *in front of* (abbreviated by −), positive numbers denote *succeeding* (abbreviated by +).

9.4.2 Maximum distance between involved entities

A strict threshold for relatedness regarding token and tree distance is proposed in Hänig & Schierle (2009). Such a threshold is valuable to filter relation candidates. According to new insights to the relation distributions, these thresholds can be defined based on statistical measures. Furthermore, a distinction between the two orders regarding both entities is essential to properly consider the observable distributions.

The threshold $maxdist^{+/-}(t_1, t_2)$ for two entity types t_1 and t_2 and a given linear order $+$ or $-$ is defined as the sum of the arithmetic mean distance between these two types and the standard deviation of the corresponding entity distribution:

$$maxdist^{+/-}_{token/tree}(t_1, t_2) = \overline{d}_{+/-}(t_1, t_2) + \sigma_{+/-}(t_1, t_2) \tag{9.2}$$

$$maxdist(e_1, e_2) = \begin{cases} maxdist^{+}(t_1, t_2) & type(e_1) = t_1 \wedge type(e_2) = t_2 \\ maxdist^{-}(t_1, t_2) & else \end{cases} \tag{9.3}$$

The influence of the automatically derived maximum distance is discussed within the evaluation part.

9.4.3 Differences between Repair Order and Internet Data

As stated before, both QUIS Corpus and AIM Corpus show significant differences in language use. These differences result in smaller impacts on entity distributions. One effect originating in diverse language use as well as longer and more complex sentences is that entities are distributed more uniformly (see Fig. 41). Due to free word-order in German, the preferred entity order cannot be observed with such a distinction as in repair order data. This effect can especially be observed on near-distance relation instances which are significantly more common in repair orders than in internet data.

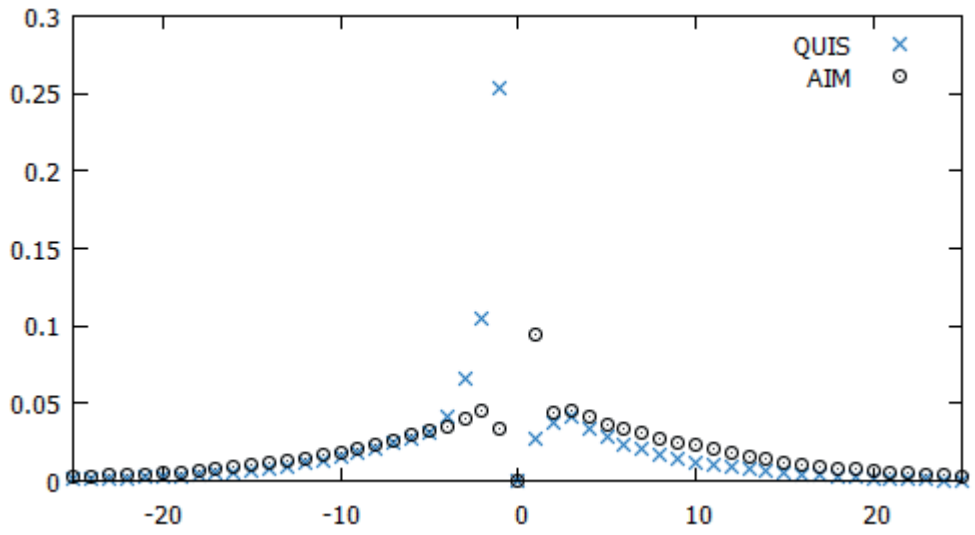


Figure 41.: Token Distributions of COMPONENT - SYMPTOM in two different corpora

Furthermore, while repair order data is relatively predictable, even completely inverse relation properties can be observed analyzing internet data. Regarding the relation COMPONENT - SYMPTOM , the preferred linear order of the entities is reversed as in the other corpus (see Fig. 41).

9.5 PROBABILISTIC MODEL

Given these observations in distributional form, several probabilities can be defined to build a probabilistic model for syntactic relation extraction. The first equation (see Equ. 9.4) gives the probability of two entities e_1 and e_2 occurring in a token or tree distance of d.

$$P_{token/tree}(e_1, e_2, d) = \frac{\left|\bigcup_{s \in S} \{(e_1, e_2) \in E_s \times E_s | \, d_{token/tree}(e_1, e_2) = d\}\right|}{\left|\bigcup_{s \in S} \{(e_1, e_2) \in E_s \times E_s\}\right|} \tag{9.4}$$

In order to consider the distance distributions regarding maximum distances, a probability is defined which is 0 for distances greater than the corresponding maximum distance:

(9.5)

$$P^{strict}_{token/tree}(e_1, e_2, d) = \begin{cases} 0 & d > maxdist_{token/tree}(e_1, e_2) \\ P_{token/tree}(e_1, e_2, d) & else \end{cases}$$

The resulting probability takes both token and tree distances of both involved entities into account and thus, is defined as:

$$P(e_1, e_2, d_{token}, d_{tree}) = P_{token}(e_1, e_2, d_{token}) \cdot P_{tree}(e_1, e_2, d_{tree}) \tag{9.6}$$

$P^{strict}(e_1, e_2, d_{token}, d_{tree})$ is analogously defined using the strict probability definitions.

Based on these probability scores, three possible strategies for RE are defined. All of them are grounded on the assumption that each entity e of the sentence's entities E is related to at least one further entity of this sentence. This premise is empirically grounded based on observations of the data. This premise may not hold for all or other domains, but the presented extraction strategies can be adapted to characteristics of other text types or domains. This point will be discussed in the conclusions.

COMPLETE EXTRACTION (CE) This extraction strategy extracts all entity pairs occurring within the respective maximum distance. Formally, a relation candidate (e_1, e_2) is considered to be a relation, if $P^{strict}(e_1, e_2, d_{token}, d_{tree}) > 0$. The probability value is not of interest for extracting relations using this strategy. Hence, it is similar to the approach presented in Hänig & Schierle (2009) where all entity pairs occurring within a maximum distance are considered to be related.

CE is suitable for extraction of multiple relations per entity, which both following strategies are not.

LOCAL PROBABILITY MAXIMIZATION (LPM) *LPM* is an iterative greedy process of successively picking the locally best candidate until no candidates remain. In detail, the steps are:

(a) Initialize the candidate set $C = \{(e_1, e_2) \in E \times E | e_1 \neq e_2\}$

(b) Pick the candidate c_r obtaining the highest probability, so that $c_r = \underset{c \in C}{\operatorname{argmax}}\ P(e_1, e_2, d_{token}(e_1, e_2), d_{tree}(e_1, e_2))$ and put it into the result set. The strict version can also be applied.

(c) Remove all candidates c from C containing any element involved in the relation c_r.

(d) Repeat steps *(b)* and *(c)* until C is empty or no candidate c achieves a zero probability.

The advantage of *LPM* over *CE* is the performance on dense entity clusters. While *CE* would extract multiple relations containing many incorrect ones *LPM* only picks the most probable candidates. The consequential disadvantage is that only one relation per entity can be extracted lowering recall when entity instances are involved in multiple relations.

GLOBAL PROBABILITY MAXIMIZATION (GPM) Similarly to *LPM* only one relation per entity is allowed when using *GPM*. Instead of picking the best local candidate, global candidate sets are evaluated. First, all possible subsets c_{set} of C are generated. Possible means that at most one entity out of E may be missing in c_{set} (in order to fulfill the initial assumption). Entities for which no relation candidate c obtaining a non-zero probability score exist, are ignored during this generation step. All elements contained in the candidate set c_{set} with the highest combined probability are considered to be relations.

The combined probability of such a candidate set is defined as:

$$P(c_{set}) = \prod_{(e_1,e_2) \in c_{set}} P(e_1, e_2, d_{token}(e_1, e_2), d_{tree}(e_1, e_2)) \tag{9.7}$$

The properties of this extraction strategy are identical to the properties of *LPM*. The only difference is that it produces solutions yielding the globally maximum probability instead of locally maximum probabilities.

9.6 EVALUATION

All three extraction strategies are evaluated on both QUIS Corpus and AIM Corpus. The results are compared to the results of current state-of-the-art algorithms (see Sec. 6.6) and incorrect classifications including over- and underpropositions are deeply analyzed. Errors originating from entity detection are ignored to solely evaluate the performance of Syntactic Relation Extraction.

9.6.1 Results

The results of the experiments are given in Table 26 for application on repair order data, and in Table 27 for internet data. The results presented in Hänig & Schierle (2009) are provided for comparison with manually defined extraction rules (*MR*).

This analysis shows that extraction based on manually generated rules performs superior than automatically created relation statistics. Nevertheless, creating highly sophisticated rules needs time and huge efforts in studying the data. The three unsupervised strategies do not need any supervision (except for the Entity Recognition step which has to be done independently from RE) and still yield reasonable results.

Complete Extraction scores slightly better regarding recall for all evaluated relation types. This observation is grounded in $maxdist$ which is greater than the manually defined distances. Consequently, more relations are extracted lowering the precision as more incorrect candidates are considered relations.

Both probability-based strategies extracted inferior amounts of relations. Restricting each entity to participate in at most one relation forbids several correct relations. Especially components often are related to locations, symptoms and corrections and in many cases, although the component is mentioned only once (e. g. LOCATION *driver side* COMPONENT *window* SYMPTOM *broken*). Such constructions cannot be easily modeled by a probabilistic model due to the available

Relation	Strategy	Precision	Recall	F-Score
COMPONENT – CORRECTION	*MR*	0.958	0.960	0.959
COMPONENT – CORRECTION	*CE*	0.932	0.968	0.950
COMPONENT – CORRECTION	*LPM*	0.881	0.873	0.877
COMPONENT – CORRECTION	*GPM*	0.887	0.889	0.888
COMPONENT – SYMPTOM	*MR*	0.910	0.941	0.925
COMPONENT – SYMPTOM	*CE*	0.896	0.949	0.922
COMPONENT – SYMPTOM	*LPM*	0.840	0.882	0.860
COMPONENT – SYMPTOM	*GPM*	0.817	0.866	0.841
COMPONENT – LOCATION	*MR*	0.964	0.954	0.959
COMPONENT – LOCATION	*CE*	0.929	0.972	0.950
COMPONENT – LOCATION	*LPM*	0.904	0.968	0.935
COMPONENT – LOCATION	*GPM*	0.899	0.956	0.927
NEGATION – SYMPTOM / CONDITION	*MR*	0.924	0.922	0.923
NEGATION – SYMPTOM / CONDITION	*CE*	0.908	0.926	0.917
NEGATION – SYMPTOM / CONDITION	*LPM*	0.798	0.861	0.828
NEGATION – SYMPTOM / CONDITION	*GPM*	0.805	0.872	0.837

Table 26.: Relation Extraction results on QUIS Corpus data

data. Appropriate probability values could be defined if sentences in the corpora would consist of such constructions only. Sentences of the QUIS Corpus do actually consist of three concatenated sentences (complaint, cause, correction) without any punctuation or other marks providing the possibility to split these parts again. Hence, statistics about with how many different entities a component normally occurs are skewed.

LPM and *GPM* do extract a higher portion of incorrect relations than both other strategies do. A probability threshold can be defined to improve extraction

accuracy, but this leads to a tradeoff between precision and recall while the overall performance does not change significantly. The results given in both tables are obtained without any special threshold. All probabilities must only be greater than 0 ($P(e_1, e_2, d_{token}(e_1, e_2), d_{tree}(e_1, e_2))$ and $P(c_{set})$ for *LPM* and *GPM*, respectively).

It can also be observed that no probability-based strategy consequentially outperforms the other one. Both show superior results on some of the studied relations and inferior on some others.

The conclusion regarding evaluation results on automotive repair order data is that simple extraction rules show superior performance. Although manually created rules outperform the identical approach with statistically grounded maximum distances on English data, *Complete Extraction* provides a reasonable strategy for application on repair order data of different languages without high losses in accuracy.

The results achieved on automotive internet data are generally inferior. This is grounded on much more complex language as presented in Sec. 5.2, especially for one of the most interesting relations COMPONENT – SYMPTOM . The argumentation is similar to the one for repair order data. Additionally, the strict concentration on relevant information cannot be observed in internet data. While the technician only records technical aspects, authors discussing technical problems do not focus in such a magnitude. Often, several ideas or assumptions are stated and thus, a lot of noise exists. Overall, *LPM* seem to outperform the other strategies (ignoring the tie for the relation COMPONENT - CORRECTION).

Internet data contains a lot of additional information – including polar utterances reflecting the sentiment of authors regarding a certain topic. These utterances are annotated on phrase level leading to an inhomogeneous distributions as different phrase types are distributed differently. A separate experiment shows, that *LPM* can detect the correct opinion holder out of several possible targets (e. g. models, components, automobile manufacturers). The evaluation data contains 150 forum entries discussing several topics. Each of the sentences contains at least one polar phrase and multiple possible opinion targets. *LPM* is

Relation	Strategy	Precision	Recall	F-Score
COMPONENT – CORRECTION	*CE*	0.387	0.812	0.524
COMPONENT – CORRECTION	*LPM*	0.439	0.783	0.563
COMPONENT – CORRECTION	*GPM*	0.439	0.783	0.563
COMPONENT – SYMPTOM	*CE*	0.155	0.557	0.243
COMPONENT – SYMPTOM	*LPM*	0.219	0.500	0.305
COMPONENT – SYMPTOM	*GPM*	0.200	0.429	0.273
COMPONENT – LOCATION	*CE*	0.573	0.931	0.709
COMPONENT – LOCATION	*LPM*	0.607	0.895	0.723
COMPONENT – LOCATION	*GPM*	0.586	0.895	0.708

Table 27.: Relation Extraction results on AIM Corpus data

employed to detect the correct opinion target on this data achieving an accuracy of 69.8%.

9.7 RELATION EXTRACTION – CONCLUSIONS

Syntactic relation extraction achieves reasonable results on both corpora and outperforms current state-of-the-art approaches (pattern-based and kernel-based) easily for automotive relations on this data.

Depending on language characteristics, different accuracy levels are achieved. Repair order data is written using restricted language, consists almost exclusively of automotive terminology and shows a very large number of relations. Complete Extraction in its original form employing manually defined extraction rules achieves the best results. The same strategy performs slightly inferior when statistically grounded maximum distances are applied instead of manually defined ones. This is important for extension to new languages for which

no sufficient language skills are available. In this use case, only the thesaurus needs to be augmented with translations to support the new language. All further processing including part-of-speech tagging, syntactic parsing and relation extraction then can be applied in an unsupervised manner.

The application on repair order data shows excellent results in the range of 80% to 95%. This level of accuracy is not achieved on automotive internet data. *Local Probability Maximization* achieves the best results. But still, the scores are significantly lower than achieved on repair order data. Especially precision values of about 20% for the important relation COMPONENT – SYMPTOM is not good enough to constitute a basis for data mining on the extracted relation data. Nevertheless, one part of the original motivation to incorporate internet data was to explore the mood and sentiment of the customers. Empirical studies proved that *LPM* is able to assign polar phrases to the corresponding opinion target scoring an accuracy of about 70%. This precision is high enough to detect tendencies and clues in the data using large scale statistics. The extracted documents then can be analyzed by an appropriate engineer who does not need to go through the complete data using this extraction mechanism as prior filtering step.

Improvements are possible especially regarding adaptation to other domains for which the premises do not hold. The evaluation results on internet data show that obviously a fair amount of entities exist that are not related to another automotive entity. Consequently, *LPM* which extracts less relations than *CE* performs better. An appropriate estimation of the ratios between all possible entity type pairs and the derivable probabilities could improve a probabilistic model. As argued before, such estimation is not possible due to the data's nature.

Kernel-based methods can be improved in several points to match the criteria demanded by RE on dirty and domain-specific data. First, kernels for RE should incorporate more knowledge about relations themselves. A relation basically consists of two entities of two types (which may be identical, e. g. company mergers). It is only possible that a sentence contains a relation instance if it contains at least two entities of the required types. The multiplication of two kernel functions still is a kernel function. This leads to the usage of an entity

type filter kernel $K_{filter}(T_1, T_2)$ in addition to the actual kernel function $K(T_1, T_2)$. The filter kernel ensures that both trees agree at least on two different entities' type or the similarity of both trees is considered zero. This proposition (see Equ. 9.8) is a more consequential approach regarding exploitation of relation characteristics than the matching function proposed by Zelenko et al. (2003).

$$K_{filter}(T_1, T_2) = \begin{cases} K(T_1, T_2) & agree\_on\_types(T_1, T_2) \\ 0 & else \end{cases} \tag{9.8}$$

Filtering all instance pairs which do not agree on at least two entities' types will increase precision significantly without influencing recall.

The accuracy of kernel-based approaches can be further improved through concentration on relevant parts of the sentence and parsetree. Training and extraction should be restricted to the lowest subtree in the constituent hierarchy containing both involved entities. This will reduce extraction of incorrect relation instances due to similar constructions in the context of the instance and not of the instance itself. Especially forum entries contain long sentences with multiple statements which often leads to overpropositions.

Unsupervised methods are essential to provide a flexible workflow which is crucial for fast extension to new text types, languages or domains. All presented kernel methods (see Sec. 4.6) rely on supervised preprocessing. Some researchers argue that supervised part-of-speech tagging can be easily replaced by unsupervised approaches (see Bod, 2006b). This may be true for grammar induction, but for the task of kernel-based relation extraction, some restrictions to this statement are necessary. Kernels exploit the syntactic structure of a sentence and tests for common tree fragments. Common tree fragments are detected by comparison of phrase types, POS tags and further properties (e. g. word types, entity annotations). Applying a comparison of POS tags and phrase types which were induced in an unsupervised manner does not yield the same results as on supervisedly annotated features as they are much more fine-grained. Hence, comparing those syntactic properties should be performed based on distributional representations of POS tags and phrase types to achieve similar results.

Part III

CONCLUSIONS

10

CONCLUSIONS

The last part of this thesis summarizes the results achieved in the previous chapters. The impact of the developed algorithms to real world applications of the automotive domain is outlined. Finally, possible directions for further research are presented.

10.1 SUMMARY

This thesis aims to develop a relation extraction algorithm to extract knowledge out of automotive data. While most approaches to RE are only evaluated on newspaper data dealing with general relations from the business world their applicability to other data sets is not well studied.

Part I of this thesis deals with theoretical foundations of information extraction algorithms. Text mining cannot be seen as the simple application of data mining methods to textual data. Instead, sophisticated methods have to be employed to accurately extract knowledge from text which then can be mined using statistical methods from the field of data mining. Information extraction itself can be divided into two subtasks: entity detection and relation extraction. The detection of entities is very domain-dependent due to terminology, abbreviations and general language use within the given domain. Thus, this task has to be solved for each domain employing thesauri or another type of lexicon. Supervised ap-

proaches to Named Entity Recognition will not achieve reasonable results unless they have been trained for the given type of data.

The task of Relation Extraction can be basically approached by pattern-based and kernel-based algorithms. The latter achieve state-of-the-art results on newspaper data and point out the importance of linguistic features. In order to analyze relations contained in textual data, syntactic features like part-of-speech tags and syntactic parses are essential. Chapter 4 presents machine learning approaches and linguistic foundations being essential for syntactic annotation of textual data and relation extraction. Chapter 6 analyzes the performance of state-of-the-art algorithms of POS tagging, syntactic parsing and relation extraction on automotive data. The findings are: supervised methods trained on newspaper corpora do not achieve accurate results when being applied on automotive data. In order to achieve acceptable results, algorithms have to be trained directly on this kind of data. As the manual annotation of data for each language and data type is too costly and inflexible, unsupervised methods are the ones to rely on.

Part II deals with the development of dedicated algorithms for all three tasks. Unsupervised POS tagging (see Chapter 7) is a well-studied task and algorithms achieving accurate tagging exist. All of them do not disambiguate high frequency words, only out-of-lexicon words are disambiguated. Most high frequency words bear syntactic information and thus, it is very important to differentiate between their different functions. Especially domain languages contain ambiguous high frequency words bearing semantic information (e. g. *pump*). In order to improve POS tagging, an algorithm for disambiguation is developed and used to enhance an existing state-of-the-art tagger. Evaluation shows that tagging accuracy is raised significantly. An approach to unsupervised syntactic parsing (see Chapter 8) is developed in order to suffice the requirements of relation extraction. These requirements include high precision results on nominal and prepositional phrases as they contain the entities being relevant for relation extraction. Furthermore, accurate shallow parsing is more desirable than deep binary parsing as it facilitates relation extraction more than deep parsing. Endocentric and exocentric constructions can be distinguished and improve proper

phrase labeling. *unsuParse* fulfills all demanded criteria and achieves competitive results on standard evaluation setups. Syntactic Relation Extraction (see Chapter 9) is an approach exploiting syntactic statistics and text characteristics to extract relations between previously annotated entities. Evaluation on two different languages and two different text types of the automotive domain shows that it achieves accurate results on repair order data. Results are less accurate on internet data, but the task of sentiment analysis and extraction of the opinion target can be mastered. Thus, the incorporation of internet data is possible and important as it provides useful insight into the customer's thoughts.

To conclude, this thesis presents a complete unsupervised workflow for relation extraction – except for the highly domain-dependent entity detection task – improving performance of each of the involved subtasks. Furthermore, this work applies NLP methods and RE approaches to real world data unveiling challenges that do not occur in high quality newspaper corpora.

10.2 APPLICATION OF THIS WORK

The Relation Extraction algorithm presented in Section 9 is integrated into quality assurance processes of an international car manufacturer. It is part of the daily data processing workflow for three years in the US and Germany. Especially in the US, a repeat repair detection mechanism based on the extracted relations (see Hänig et al., 2010, 2011) is used to improve fraud detection and the online appointment system. Statistics about which repair actions reveal the repair approach yielding the best success and influence the spare part ordering process before the customer actually comes in for the repair.

The presented approaches are evaluated in terms of what information can be extracted in an automatic way from the growing amount of textual data available in automotive internet fora. In this use case, the sentiment analysis is of special interest as it unveils the opinions of the customers regarding all manufacturers (see Bank & Hänig, 2011).

10.3 POTENTIAL FOR FURTHER RESEARCH

Regarding all three tasks – POS tagging, syntactic parsing and relation extraction – several improvements are possible are extensively discussed in the respective sections. Thus, only a short comprehension is given at this point.

For POS tagging, the most promising improvements are a more sophisticated integration of part-of-speech disambiguation and the inclusion of morphology.

Possibilities for unlexicalized unsupervised syntactic parsing are nearly completely exhausted. The task of phrase labeling can be further improved by better exploitation of the parts-of-speech contained in a phrase. Apart from that, partly lexicalization of models should be explored as full lexicalization is not possible due to data-sparseness problems.

Relation Extraction on arbitrary relation types is a young research field as most approaches only target simple semantic relations. Ideas from syntactic relation extraction presented in this thesis should be combined with kernel methods (or encoded as kernel function) to improve relation extraction results on arbitrary logical relations.

BIBLIOGRAPHY

Abend, O., Reichart, R., & Rappoport, A. (2010). Improved Unsupervised POS Induction through Prototype Discovery. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics* (pp. 1298—-1307). Uppsala, Sweden: Association for Computational Linguistics.

Abney, S. P. (1989). A computational model of human parsing. *Journal of Psycholinguistic Research*, 18(1), 129–144.

Adger, D. (2003). *Core Syntax: A Minimalist Approach*. Oxford University Press.

Agichtein, E. & Gravano, L. (2000). Snowball: Extracting Relations from Large Plain-Text Collections. In *Proceedings of the Fifth {ACM} International Conference on Digital Libraries*.

Aizerman, M. A., Braverman, E., & Rozonoer, L. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25, 821–837.

Alfonseca, E., Ruiz-Casado, M., Okumura, M., & Castells, P. (2006). Towards Large-scale Non-taxonomic Relation Extraction : Estimating the Precision of Rote Extractors. In *Proceedings of the 2nd Workshop on Ontology Learning and Population*, number July (pp. 49–56). Sydney, Australia.

Appelt, D. E., Bear, J., Hobbs, J. R., Israel, D., & Tyson, M. (1992). SRI International FASTUS system: MUC-4 test results and analysis. In *MUC4 '92: Proceedings of the 4th conference on Message understanding* (pp. 143–147). Morristown, NJ, USA: Association for Computational Linguistics.

Auger, A. & Barrière, C. (2008). Pattern-based approaches to semantic relation extraction : A state-of- the-art. *Terminology*, 14(1), 1–19.

Baldewein, U., Erk, K., Padó, S., & Prescher, D. (2004). Semantic Role Labelling with Similarity-Based Generalization Using EM-based Clustering. In *Proceedings of the third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, number July (pp. 1–7). Barcelona, Spain.

Bank, M. & Hänig, C. (2011). Using the Internet as Sensor for Customer Perception. In *Proceedings of the Fachtagung zum Text- und Data Mining für die Qualitätsanalyse in der Automobilindustrie* (pp. 49–55). Leipzig, Germany.

Barla, A., Franceschi, E., Odone, F., & Verri, A. (2002). Image Kernels. In *Pattern Recognition with Support Vector Machines* (pp. 617–628).

Baum, L. E. (1972). An Inequality and Associated Maximization Technique in Statistical Estimation of Probabilistic Functions of a Markov Process. *Inequalities*, 3, 1–8.

Baum, L. E. & Eagon, J. A. (1967). An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bulletin of the American Mathematical Society*, 73, 360–363.

Baum, L. E. & Petrie, T. (1966). Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *The Annals of Mathematical Statistics*, 37(6), 1554–1563.

Baum, L. E., Petrie, T., Soules, G., & Weiss, N. (1970). A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *The Annals of Mathematical Statistics*, 41(1), 164–171.

Baum, L. E. & Sell, G. R. (1968). Growth transformations for functions on manifolds. *Pacific Journal of Mathematics*, 27(2), 211–227.

Bayes, T. & Price, R. (1763). An Essay towards Solving a Problem in the Doctrine of Chances. *Philosophical Transactions of the Royal Society of London*, 53, 370–418.

Berland, M. & Charniak, E. (1999). Finding Parts in Very Large Corpora. In *ACL '99 Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, volume 1910 (pp. 57–64).

Biemann, C. (2006a). Chinese Whispers: an Efficient Graph Clustering Algorithm and its Application to Natural Language Processing Problems. In *Proceedings of TextGraphs: the First Workshop on Graph Based Methods for Natural Language Processing on the First Workshop on Graph Based Methods for Natural Language Processing* (pp. 73—-80). New York, USA: Association for Computational Linguistics.

Biemann, C. (2006b). Unsupervised part-of-speech tagging employing efficient graph clustering. In *Proceedings of the COLING/ACL 2006 Student Research Workshop* (pp. 73–80). Sydney, Australia: Association for Computational Linguistics.

Biemann, C. & Bordag, S. (2003). {L}ernen paradigmatischer {R}elationen auf iterierten {K}ollokationen. In *Anwendungen des deutschen Wortnetzes in Theorie und Praxis*.

Biemann, C., Bordag, S., Heyer, G., Quasthoff, U., & Wolff, C. (2004). Language-Independent Methods for Compiling Monolingual Lexical Data. In A. Gelbukh (Ed.), *Computational Linguistics and Intelligent Text Processing*, volume 2945 of *Lecture Notes in Computer Science* (pp. 217–228). Springer Berlin / Heidelberg.

Black, E., Abney, S., Flickenger, S., Gdaniec, C., Grishman, C., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J., Liberman, M., Marcus, M., Roukos, S., Santorini, B., & Strzalkowski, T. (1991). Procedure for quantitatively comparing the syntactic coverage of English grammars. In E. Black (Ed.), *Proceedings of the workshop on Speech and Natural Language*, number VPAST did (pp. 306–311). Pacific Grove, USA: Association for Computational Linguistics.

Blohm, S. & Cimiano, P. (2007). Using the Web to Reduce Data Sparseness in Pattern-Based Information Extraction. In J. Kok, J. Koronacki, R. de Mantaras, S. Matwin, D. Mladenic, & A. Skowron (Eds.), *Knowledge Discovery in*

Databases: PKDD 2007, volume 4702 of *Lecture Notes in Computer Science* (pp. 18–29). Springer Berlin / Heidelberg.

Bloomfield, L. (1933). *Language*. Chicago, USA: University of Chicago Press.

Bod, R. (1998). *Beyond Grammar: An experience-based theory of language*. CSLI Publications. Cambridge University Press.

Bod, R. (2006a). An all-subtrees approach to unsupervised parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, volume 44 (pp. 865–872). Sydney, Australia: Association for Computational Linguistics.

Bod, R. (2006b). Unsupervised parsing with U-DOP. In *Proceedings of the Tenth Conference on Computational Natural Language*, number June (pp. 85–92). New York, USA: Association for Computational Linguistics.

Bod, R. (2007a). A linguistic investigation into unsupervised DOP. *Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition - CACLA '07*, (June), 1–8.

Bod, R. (2007b). Is the End of Supervised Parsing in Sight? In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, volume 45 (pp. 400—-407). Prague, Czech Republic: Association for Computational Linguistics.

Bordag, S. (2007). *Elements of Knowledge-free and Unsupervised Lexical Acquisition*. Phd, University of Leipzig.

Borensztajn, G. & Zuidema, W. (2007). *Bayesian Model Merging for Unsupervised Constituent Labeling and Grammar Induction*. Technical report.

Borthwick, A., Sterling, J., Agichtein, E., & Grishman, R. (1998). Exploiting Diverse Knowledge Sources via Maximum Entropy in Named Entity Recognition. In *Proceedings of the sixth workshop on very large corpora* (pp. 152–160).

Bottou, L. & Bengio, Y. (1995). Convergence Properties of the K-Means Algorithms. In *Advances in Neural Information Processing Systems 7* (pp. 585–592).

Brants, S., Dipper, S., Hansen, S., Lezius, W., & Smith, G. (2002). The TIGER Treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories Sozopol*.

Brants, T. (2000). TnT - a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied natural language processing*, number i (pp. 224—-231). Seattle, Washington: Association for Computational Linguistics.

Brin, S. (1998). Extracting Patterns and Relations from the World Wide Web. In *WebDB Workshop at 6th International Conference on Extending Database Technology, {EDBT}'98*.

Brito, M. R., Chávez, E. L., Quiroz, A. J., & Yukich, J. E. (1997). Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection. *Statistics & Probability Letters*, 35(1), 33–42.

Büchler, M. (2006). *Flexibles Berechnen von Kookkurrenzen auf strukturierten und unstrukturierten Daten*. PhD thesis, University of Leipzig.

Bunescu, R. C. & Mooney, R. J. (2005). A Shortest Path Dependency Kernel for Relation Extraction. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing - HLT '05*, number October (pp. 724–731). Morristown, NJ, USA: Association for Computational Linguistics.

Caraballo, S. (1999). Automatic acquisition of a hypernym-labeled noun hierarchy from text. In *Proceedings of ACL-99* (pp. 120–126).

Carroll, J., Minnen, G., & Briscoe, T. (1999). Corpus Annotation for Parser Evaluation. In *Proceedings of the EACL workshop on Linguistically Interpreted Corpora*, number June (pp. 35–41). Bergen, Norway: Association for Computational Linguistics.

Chang, C.-C. & Lin, C.-J. (2011). {LIBSVM}: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3), 27:1—-27:27.

Chang, N. (2004). Putting Meaning into Grammar Learning. In *COLING 2004 Psycho-Computational Models of Human Language Acquisition* (pp. 19–26). Geneva, Switzerland: Association for Computational Linguistics.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, number c (pp. 132—-139). Seattle, Washington: Morgan Kaufmann Publishers Inc.

Chieu, H. L. & Ng, H. T. (2002). Named entity recognition: a maximum entropy approach using global information. In *Proceedings of the 19th international conference on Computational linguistics* (pp. 1–7). Morristown, NJ, USA: Association for Computational Linguistics.

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory*, 2(3), 113–124.

Chomsky, N. (1957). *Syntactic Structures*. The Hague / Paris: Mouton.

Chomsky, N. (1963). Formal properties of grammars. In *Handbook of Mathematical Psychology Vol. 2* (pp. 323–418). John Wiley and Sons Ltd.

Chomsky, N. & Halle, M. (1968). *The Sound Pattern of English*. Studies in language. Harper & Row.

Cimiano, P., Pivk, A., & Steffen, L. S.-t. (2003a). Learning Taxonomic Relations from Heterogeneous Sources of Evidence. In *Ontology Learning from Text: Methods, Evaluation and Applications* (pp. 55–73). Amsterdam, Netherlands: IOS Press.

Cimiano, P., Staab, S., & Tane, J. (2003b). Automatic acquisition of taxonomies from text: FCA meets NLP. In *Proceedings of the International Workshop & Tu-*

torial on Adaptive Text Extraction and Mining, number September (pp. 10–17). Cavtat-Dubrovnik, Croatia.

Clark, A. (2003). Combining distributional and morphological information for part of speech induction. In *Proceedings of the EACL* (pp. 59—-66). Budapest, Hungary: Association for Computational Linguistics.

Collins, M. (2002). Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, EMNLP '02 (pp. 1–8). Stroudsburg, PA, USA: Association for Computational Linguistics.

Collins, M. (2003). Head-Driven Statistical Models for Natural Language Parsing. *Computational Linguistics*, 29(4), 589–637.

Collins, M. & Duffy, N. (2001). Convolution Kernels for Natural Language. In *Proceedings of the 2001 Neural Information Processing Systems Conference* Vancouver, Canada.

Collins, M. & Singer, Y. (1999). Unsupervised Models for Named Entity Classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora* (pp. 100–110).

Collins, M. J. (1996). A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics* (pp. 184–191). Santa Cruz, USA: Association for Computational Linguistics.

Cormen, T. H., Stein, C., Leiserson, C. E., & Rivest, R. L. (2001). *Introduction to Algorithms*. B&T.

Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273–297.

Cristianini, N. & Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press.

Crocker, M. W., Pickering, M., & Clifton, C., Eds. (1999). *Architectures and Mechanisms for Language Processing*. New York, NY, USA: Cambridge University Press.

Culotta, A. & Sorensen, J. (2004). Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics - ACL '04* Morristown, NJ, USA: Association for Computational Linguistics.

Cutting, D., Kupiec, J., Pedersen, J., & Sibun, P. (1992). A practical part-of-speech tagger. In *Proceedings of the third conference on Applied natural language processing* (pp. 133—-140). Trento, Italy: Association for Computational Linguistics.

De Morgan, A. (1966). *On the syllogism and other logical writings*. Routledge.

de Saussure, F. (1966). *Grundlagen der allgemeinen Sprachwissenschaft*. Berlin, Germany: De Gruyter.

Dice, L. R. (1945). Measures of the Amount of Ecologic Association Between Species. *Ecology*, 26(3), 297–302.

Duan, K.-B. & Keerthi, S. S. (2005). Which Is the Best Multiclass SVM Method ? An Empirical Study. In *Multiple Classifier Systems* (pp. 732–760). Berlin, Germany: Springer Berlin / Heidelberg.

Duchier, D. (1999). Axiomating dependency parsing using set constraints. In *Sixth Meeting on Mathematics of Language* Orlando, USA.

Dunning, T. (1993). Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics*, 19(1), 61–74.

Edwards, A. W. F. (1972). *Likelihood: an account of the statistical concept of likelihood and its applications to scientific inference*. Cambridge University Press.

Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A.-M., Shaked, T., Soderland, S., Weld, D. S., & Yates, A. (2004). Web-scale information extraction in

KnowItAll. In *Proceedings of the 13th International World Wide Web Conference* (pp. 100–110).

Etzioni, O., Cafarella, M., Downey, D., Popescu, A., Shaked, T., Soderland, S., Weld, D., & Yates, A. (2005). Unsupervised named-entity extraction from the Web: An experimental study. *Artificial Intelligence*, 165(1), 91–134.

Evert, S. (2004). *The Statistics of Word Cooccurrences Word Pairs and Collocations*. PhD thesis, Universitï¿½t Stuttgart.

Feldman, R. & Rosenfeld, B. (2006). Boosting unsupervised relation extraction by using NER. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, number July (pp. 473—-481). Sydney, Australia: Association for Computational Linguistics.

Feldman, R. & Sanger, J. (2007). *The Text Mining Handbook*. Cambridge University Press.

Firth, J. R. (1957). A synopsis of linguistic theory 1930-1955. *Studies in Linguistic Analysis*, (pp. 1–32).

Forney Jr., G. D. (1973). The Viterbi Algorithm. *Proceedings of the IEEE*, 61(3), 268—-278.

Fundel, K., Küffner, R., Zimmer, R., Informatik, I., & München, L.-m.-u. (2006). RelEx - Relation extraction using dependency parse trees. *Bioinformatics*, 23, 1–8.

Gao, J. & Johnson, M. (2008). A comparison of Bayesian estimators for unsupervised Hidden Markov Model POS taggers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08 (pp. 344–352). Stroudsburg, PA, USA: Association for Computational Linguistics.

Gao, L., Chang, E., & Han, S. (2005). Powerful Tool to Expand Business Intelligence : Text Mining. In *Transactions on ENFORMATIKA, Systems Sciences and Engineering-ESSE* (pp. 110–115).

Gazdar, G., Klein, E., Pullum, G. K., & Sag, I. A. (1985). *Generalized Phrase Structure Grammar*. Harvard University Press.

Gildea, D. & Jurafsky, D. (2002). Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28(3), 245–288.

Girju, R., Badulescu, A., Moldovan, D., & Winston, H. (2003). Learning Semantic Constraints for the Automatic Discovery of Part-Whole Relations. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1* (pp. 1–8).

Goldberg, A. E. (2006). *Constructions at Work*. Oxford, England: Oxford University Press.

Goldwater, S. & Griffiths, T. L. (2007). A Fully Bayesian Approach to Unsupervised Part-of-Speech Tagging. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, number June (pp. 744–751). Prague, Czech Republic.

Graça, J. a. V., Ganchev, K., Taskar, B., & Pereira, F. (2009). Posterior vs. Parameter Sparsity in Latent Variable Models. In *Neural Information Processing Systems Conference* (pp. 1–9).

Grishman, R. (1995). The NYU system for MUC-6 or where's the syntax? In *MUC6 '95: Proceedings of the 6th conference on Message understanding* (pp. 167–175). Morristown, NJ, USA: Association for Computational Linguistics.

Grishman, R., Macleod, C., & Sterling, J. (1992). Evaluating parsing strategies using standardized parse files. In *Proceedings of the third conference on Applied natural language processing* - (pp. 156–161). Trento, Italy: Association for Computational Linguistics.

Grishman, R. & Sundheim, B. (1995). Design of the MUC-6 evaluation. In *MUC6 '95: Proceedings of the 6th conference on Message understanding* (pp. 1–11). Morristown, NJ, USA: Association for Computational Linguistics.

Gunning, R. (1952). *The Technique of Clear Writing*. McGraw-Hill.

Haghighi, A. & Klein, D. (2006). Prototype-driven grammar induction. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, number July (pp. 881–888). Sydney, Australia: Association for Computational Linguistics.

Hamp, B. & Feldweg, H. (1997). GermaNet - a Lexical-Semantic Net for German. In *Proceedings of ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*.

Hänig, C. (2010a). Improvements in Unsupervised Co-Occurrence Based Parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, number July (pp. 1–8). Uppsala, Sweden: Association for Computational Linguistics.

Hänig, C. (2010b). Unsupervised Part-of-Speech Disambiguation for High Frequency Words and Its Influence on Unsupervised Parsing. In *Proceedings of the 11th International Conference on Intelligent Text Processing and Computational Linguistics* (pp. 113–120). Iasi, Romania: Springer.

Hänig, C. (2011). Knowledge-free Verb Detection through Tag Sequence Alignment. In *Proceedings of the 18th International Nordic Conference of Computational Linguistics* (pp. 291–294). Riga, Latvia.

Hänig, C., Bordag, S., & Quasthoff, U. (2008). UnsuParse: Unsupervised Parsing with unsupervised Part of Speech tagging. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation* (pp. 1109–1114). Marrakech, Morocco: European Language Resources Association (ELRA).

Hänig, C. & Schierle, M. (2009). Relation Extraction based on Unsupervised Syntactic Parsing. In *Proceedings of the Conference on Text Mining Services* (pp. 65–70). Leipzig, Germany.

Hänig, C., Schierle, M., & Trabold, D. (2010). Comparison of Structured vs. Unstructured Data for Industrial Quality Analysis. In *Proceedings of the World*

Congress on Engineering and Computer Science 2010, volume I (pp. 432–438). San Francisco, USA: IAENG.

Hänig, C., Schierle, M., & Trabold, D. (2011). Benefits of Unstructured Data for Industrial Quality Analysis. In *Intelligent Automation and Systems Engineering* chapter 20, (pp. 257–270). Springer.

Harris, Z. (1954). *Distributional Structure*. Chicago, USA: University of Chicago Press.

Harris, Z. S. (1951). *Methods in Structural Linguistics*. Chicago, USA: University of Chicago Press.

Harris, Z. S. (1968). *Mathematical Structure of Language*. Krieger Pub Co.

Haspelmath, M., Dryer, M. S., & Gil, D. (2005). *The world atlas of language structures*. Oxford Linguistics.

Hatzivassiloglou, V. & Mckeown, K. R. (1997). Predicting the semantic orientation of adjectives. *Proceedings of the 35th annual meeting on Association for Computational Linguistics -*, (pp. 174–181).

Haussler, D. (1999). *Convolution Kernels on Discrete Structures*. Technical report, University of California, Santa Cruz.

Hearst, M. A. (1992). *Automatic Acquisition of Hyponyms from Large Text Corpora*. Technical Report S2K-92-09, University of California, Berkeley.

Heyer, G. & Bordag, S. (2007). A Structuralist Framework for Quantitative Linguistics. In *Aspects of Automatic Text Analysis / Series: Studies in Fuzziness and Soft Computing*. Berlin, New York: Springer.

Heyer, G., Quasthoff, U., & Wittig, T. (2006). *Text Mining: Wissensrohstoff Text*. Herdecke [u.a.]: W3L-Verl.

Holmes, D. (1994). Authorship attribution. *Computers and the Humanities*, 28(2), 87–106.

Isozaki, H. & Kazawa, H. (2002). Efficient support vector classifiers for named entity recognition. In *Proceedings of the 19th international conference on Computational linguistics* (pp. 1–7). Morristown, NJ, USA: Association for Computational Linguistics.

Jaccard, P. (1901). Distribution comparée de la flore alpine dans quelques régions des Alpes occidentales et orientales . *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37, 547–579.

Jackendoff, R. (1977). *X-Bar Syntax: A Study of Phrase Structure*. Cambridge, USA: MIT Press.

Jackson, P. & Moulinier, I. (2002). Natural language processing for online applications. Text retrieval, extraction and categorization. *Natural Language Processing*, 5.

Jiang, J. & Zhai, C. (2007). A Systematic Exploration of the Feature Space for Relation Extraction. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference* (pp. 113–120). Rochester, New York: Association for Computational Linguistics.

Kazama, J., Makino, T., Ohta, Y., & Tsujii, J. (2002). Tuning Support Vector Machines for Biomedical Named Entity Recognition. In *Proceedings of the ACL-02 Workshop on Natural Language Processing in the Biomedical Domain* (pp. 1–8).

Kazama, J. & Torisawa, K. (2007). Exploiting Wikipedia as External Knowledge for Named Entity Recognition. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, number March (pp. 698–707). Prague, Czech Republic: Association for Computational Linguistics.

Kim, S.-M. & Hovy, E. (2004). Determining the sentiment of opinions. *Proceedings of the 20th international conference on Computational Linguistics - COLING '04*, (pp. 1367–es).

Kim, S.-m. & Hovy, E. (2005). Automatic Detection of Opinion Bearing Words and Sentences. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing (IJCNLP)* (pp. 61–66).

Klein, D. (2005). *The Unsupervised Learning of Natural Language Structure*. Phd, Stanford University.

Klein, D. & Manning, C. D. (2002a). A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, volume 14 (pp. 128–135). Philadelphia, USA: MIT Press.

Klein, D. & Manning, C. D. (2002b). Natural Language Grammar Induction using a Constituent-Context Model. *Advances in Neural Information Processing Systems*, 14, 35–42.

Klein, D. & Manning, C. D. (2003a). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1* (pp. 423—-430). Sapporo, Japan: Association for Computational Linguistics.

Klein, D. & Manning, C. D. (2003b). Fast Exact Inference with a Factored Model for Natural Language Parsing. In *Advances in Neural Information Processing Systems 15* (pp. 3—-10).: MIT Press.

Klein, D. & Manning, C. D. (2004). Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics* Barcelona, Spain: Association for Computational Linguistics.

Kozareva, Z. (2006). Bootstrapping Named Entity Recognition with Automatically Generated Gazetteer Lists. In *Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop* (pp. 15–21). Trento, Italy: Association for Computational Linguistics.

Krauthammer, M. & Nenadic, G. (2004). Term identification in the biomedical literature. *J. of Biomedical Informatics*, 37(6), 512–526.

Kripke, S. A. (1980). *Naming and Necessity*. Harvard university Press.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2), 83–97.

Lamar, M., Maron, Y., & Bienenstock, E. (2010a). Latent-Descriptor Clustering for Unsupervised POS Induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, number October (pp. 799–809).

Lamar, M., Maron, Y., Johnson, M., & Bienenstock, E. (2010b). SVD and Clustering for Unsupervised POS Tagging. In *Proceedings of the ACL 2010 Conference Short Papers* (pp. 215—-219). Uppsala, Sweden: Association for Computational Linguistics.

Larsen, B. & Aone, C. (1999). Fast and effective text mining using linear-time document clustering. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '99 (pp. 16–22). New York, NY, USA: ACM.

Lehmann, M. (2006). *{I}mplementierung eines {T}ext-{M}ining-{S}ystems auf {B}asis des {U}{I}{M}{A}-{F}rameworks*. Master, Universität Leipzig.

Li, W. X. (1996). Building efficient incremental LL parsers by augmenting LL tables and threading parse trees. *Comput. Lang.*, 22(4), 225–235.

Lin, D. (1996). Dependency-based parser evaluation: a study with a software manual corpus. In R. Sutcliffe, H.-D. Koch, & A. McElligott (Eds.), *Industrial Parsing of Software Manuals* chapter 2, (pp. 13–24). Amsterdam, Netherlands: Rodopi.

Lloyd, S. P. (1957). *Least square quantization in PCM*. Technical Report 2, Bell Telephone Laboratories.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., & Watkins, C. (2002). Text Classification using String Kernels. *Journal of Machine Learning Research*, 2(3), 419–444.

MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability* (pp. 281–297).

MacWhinney, B. (2000). *The CHILDES project: Tools for analyzing talk*. Lawrence Erlbaum Assoc Inc.

Mahn, M. (2005). {E}valuation und {V}erknï¿½pfung von {K}ookkurrenzen hï¿½herer {O}rdnung und {P}robabilistic {L}atent {S}emantic {A}nalysis. Master's thesis, Leipzig University.

Manning, C. D. & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*, volume 8. New York, USA: MIT Press.

Marcus, M. P., Santorini, B., & Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2), 313—-330.

Markov, A. A. (1954). The Theory of Algorithms. *Trudy Mat. Inst. Steklov.*, 42, 3–375.

McCallum, A. & Li, W. (2003). Early Results for Named Entity Recognition with Conditional Random Fields , Feature Induction and Web-Enhanced Lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4* (pp. 188–191).: Association for Computational Linguistics.

Mehler, A. & Wolff, C. (2005). Einleitung: Perspektiven und Positionen des Text Mining [Einf{ü}hrung in das Themenheft Text Mining des LDV-Forum]. *LDV-Forum*, 20(1), 1–18.

Meila, M. (2007). Comparing clusterings - an information based distance. *Journal of Multivariate Analysis*, 98(5), 873–895.

Mercer, J. (1909). Functions of Positive and Negative Type, and Their Connection with the Theory of Integral Equations. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 83(559), 69–70.

Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., & Miller, K. J. (1990). Introduction to {WordNet}: {A}n On-line Lexical Database. *Int J Lexicography*, 3(4), 235–244.

Miller, S., Fox, H., Ramshaw, L., & Weischedel, R. (2000). A Novel Use of Statistical Parsing to Extract Information from Text. In *Proceedings of the 6th Applied Natural Language Processing Conference* (pp. 226–232).: Association for Computational Linguistics.

Moilanen, K. & Pulman, S. (2007). Sentiment Composition. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP)*.

Moschitti, A. (2004). A study on convolution kernels for shallow semantic parsing. *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics - ACL '04*, (pp. 335–es).

Moschitti, A. (2006). Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees. In *Proceedings of the 17th European Conference on Machine Learning* (pp. 318–329). Berlin, Germany.

Moschitti, A., Pighin, D., & Basili, R. (2006). Semantic role labeling via tree kernel joint inference. *Proceedings of the Tenth Conference on Computational Natural Language Learning - CoNLL-X '06*, (pp.61).

Munkres, J. (1957). Algorithms for the Assignment and Transportation Problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1), 32–38.

Nadeau, D., Turney, P. D., & Matwin, S. (2006). Unsupervised Named-Entity Recognition: Generating Gazetteers and Resolving Ambiguity. In L. Lamon-

tagne & M. Marchand (Eds.), *Canadian Conference on AI*, volume 4013 of *Lecture Notes in Computer Science* (pp. 266–277).: Springer.

Neviarouskaya, A., Prendinger, H., & Ishizuka, M. (2009). Compositionality Principle in Recognition of Fine-grained Emotions from Text. In *Proceedings of the 3rd International Conference on Weblogs and Social Media (ICWSM)* (pp. 278–281).

Nock, R. & Nielsen, F. (2006). On Weighting Clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(8), 1223–1235.

Osborne, T., Putnam, M., & Gross, T. M. (2011). Bare phrase structure, labelless trees, and specifier-less syntax. Is Minimalism becoming a dependency grammar? *The Linguistic Review*, 28(3), 315–364.

Pagallo, G. M. (1990). *Adaptative decision tree algorithms for learning from examples*.

Palmer, M., Gildea, D., & Kingsbury, P. (2005). The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1), 71–106.

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? Sentiment Classification using Machine Learning Techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 79—-86). Philadelphia, USA.

Pantel, P., Rey, M., Pennacchiotti, M., Disp, A. R. T. G., & Politecnico, V. (2006). Espresso : Leveraging Generic Patterns for Automatically Harvesting Semantic Relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, number Hindle 1990 (pp. 113–120).

Pasca, M. (2005). Finding Instance Names and Alternative Glosses on the Web: WordNet Reloaded. In A. Gelbukh (Ed.), *Computational Linguistics and Intelligent Text Processing*, volume 3406 of *Lecture Notes in Computer Science* (pp. 280–292). Springer Berlin / Heidelberg.

Pecina, P. (2005). An extensive empirical study of collocation extraction methods. In *Proceedings of the ACL Student Research Workshop*, volume 100 (pp. 13—-18). Ann Arbor, USA: Association for Computational Linguistics.

Pecina, P. & Schlesinger, P. (2006). Combining association measures for collocation extraction. In *Proceedings of the COLING/ACL on Main conference poster sessions*, COLING-ACL '06 (pp. 651–658). Stroudsburg, PA, USA: Association for Computational Linguistics.

Pennacchiotti, M., Disp, A. R. T. G., Politecnico, V., Pantel, P., & Rey, M. (2006). Ontologizing Semantic Relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, number July (pp. 793–800).

Petasis, G., Paliouras, G., & Karkaletsis, V. (2004). E-grids: Computationally efficient grammatical inference from positive examples.

Polanyi, L. & Zaenen, A. (2006). Contextual Valence Shifters. In *Computing Attitude and Affect in Text: Theory and Application* (pp. 1–9).

Pollard, C. & Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar*. University Of Chicago Press.

Pollard, D. (1982). A Central Limit Theorem for k-Means Clustering. *Annals of Probability*, 10(4), 919–926.

Ponvert, E., Baldridge, J., & Erk, K. (2011). Simple Unsupervised Grammar Induction from Raw Text with Cascaded Finite State Models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, number 1999 (pp. 1077–1086). Portland, USA: Association for Computational Linguistics.

Post, E. L. (1943). Formal Reductions of the General Combinatorial Decision Problem. *American Journal of Mathematics*, 65(2), 197–215.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). *Numerical Recipes*. Cambridge University Press, 3rd editio edition.

Quasthoff, U., Richter, M., & Biemann, C. (2006). Corpus portal for search in monolingual corpora. In *Proceedings of the fifth international conference on Language Resources and Evaluation* (pp. 1799—-1802). Genova, Italy: European Language Resources Association (ELRA).

Radford, A. (1988). *Transformational Grammar*. Cambridge, USA: Cambridge University Press.

Ravichandran, D. & Hovy, E. (2002). Learning Surface Text Patterns for a Question Answering System. In *ACL '02 Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, number July (pp. 41–47).: Association for Computational Linguistics.

Reichart, R. & Rappoport, A. (2008). Unsupervised induction of labeled parse trees by clustering with syntactic features. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, number August (pp. 721—-728). Manchester, United Kingdom: Association for Computational Linguistics.

Reichart, R. & Rappoport, A. (2009). The NVI Clustering Evaluation Measure. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, number June (pp. 165—-173). Boulder, USA: Association for Computational Linguistics.

Remus, R. & Hänig, C. (2011). Towards Well-grounded Phrase-level Polarity Analysis. In *Proceedings of the 12th International Conference on Intelligent Text Processing and Computational Linguistics* (pp. 380–392). Tokyo, Japan: Springer.

Remus, R., Quasthoff, U., & Heyer, G. (2010). SentiWS - a Publicly Available German-language Resource for Sentiment Analysis. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC)* (pp. 1168–1171).

Riloff, E. (1996). Automatically Generating Extraction Patterns from Untagged Text. *AAAI/IAAI2*, 2, 1044–1049.

Riloff, E. & Thelen, M. (2002). A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language processing*.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14(5), 465–471.

Roark, B. (2001). Probabilistic Top-Down Parsing and Language Modeling. *Computational Linguistics*, 27(2), 249–276.

Robinson, J. J. (1970). Dependency structures and transformation rules. *Language*, 46(2), 259–285.

Rosenberg, A. & Hirschberg, J. (2007). {V}-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, number June (pp. 410–420).: Association for Computational Linguistics.

Rosenfeld, B. & Feldman, R. (2006a). High-Performance Unsupervised Relation Extraction from Large Corpora. In *Proceedings of the Sixth International Conference on Data Mining* (pp. 1032—-1037). Hong Kong, China: IEEE Computer Society.

Rosenfeld, B. & Feldman, R. (2006b). URES : an Unsupervised Web Relation Extraction System. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, number July (pp. 667–674). Sydney, Australia: Association for Computational Linguistics.

Rosenfeld, B. & Feldman, R. (2007). Clustering for Unsupervised Relation Identification. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management* (pp. 411—-418). Lisbon, Portugal: Association for Computing Machinery.

Sang, E. F. T. K. (2002). Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of CoNLL-2002* (pp. 155–158).: Taipei, Taiwan.

Sang, E. F. T. K. & Meulder, F. D. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *In Walter Daelemans and Miles Osborne, editors, Proceedings of CoNLL-2003*, volume pages (pp. 142–147).

Santamaría, J. & Araujo, L. (2010). Identifying Patterns for Unsupervised Grammar Induction. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, number July (pp. 38–45). Uppsala, Sweden: Association for Computational Linguistics.

Santorini, B. (1990). *Part-of-speech tagging guidelines for the Penn Treebank Project*. Technical report, University of Pennsylvania.

Schierle, M. (2011). *Language Engineering for Information Extraction*. Phd, University of Leipzig.

Schierle, M. & Schulz, S. (2007). Bootstrapping Algorithms for an Application in the Automotive Domain. In *Proceedings of the Sixth International Conference on Machine Learning and Applications*, ICMLA '07 (pp. 198–203). Washington, DC, USA: IEEE Computer Society.

Schierle, M. & Trabold, D. (2008). Extraction of Failure Graphs from Structured and Unstructured Data. In *Proceedings of the 2008 Seventh International Conference on Machine Learning and Applications* (pp. 324–330). Washington, DC, USA: IEEE Computer Society.

Schierle, M. & Trabold, D. (2010). Multilingual Knowledge-Based Concept Recognition in Textual Data. In H. H. Bock, W. Gaul, M. Vichi, P. Arabie, D. Baier, F. Critchley, R. Decker, E. Diday, M. Greenacre, C. Lauro, J. Meulman, P. Monari, S. Nishisato, N. Ohsumi, O. Opitz, G. Ritter, M. Schader, C. Weihs, A. Fink, B.

Lausen, W. Seidel, & A. Ultsch (Eds.), *Advances in Data Analysis, Data Handling and Business Intelligence*, Studies in Classification, Data Analysis, and Knowledge Organization (pp. 327–336). Berlin, Heidelberg: Springer Berlin Heidelberg.

Schiller, A., Teufel, S., Stöckert, C., & Thielen, C. (1995). *Guidelines für das Tagging deutscher Textcorpora mit STTS*. Technical report, University of Stuttgart / University of Tübingen.

Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, volume 12 Manchester, UK.

Schnitzler, S. & Eitrich, T. (2006). *{E}ine {E}inführung zu {S}tring-{K}ernen für die {S}equenzanalyse mit {S}upport-{V}ektor-{M}aschinen und anderen {K}ern-basierten {L}ernalgorithmen*. Technical report, Forschungszentrum Jï¿½lich GmbH, Zentralinstitut fï¿½r Angewandte Mathematik.

Schütze, H. (1995). Distributional Part-of-Speech Tagging. In *Proceedings of the seventh conference on European chapter of the Association for Computational Linguistics* (pp. 141—-148). Dublin, Ireland: Morgan Kaufmann Publishers Inc.

Schütze, H. & Walsh, M. (2008). A graph-theoretic model of lexical syntactic acquisition. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, number October (pp. 917—-926). Honolulu, Hawaii: Association for Computational Linguistics.

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

Settles, B. (2004). Biomedical Named Entity Recognition Using Conditional Random Fields and Rich Feature Sets. In *Proceedings of the International Joint Work-*

shop on Natural Language Processing in Biomedicine and its Applications (pp. 104–107). Geneva, Switzerland: Association for Computational Linguistics.

Shannon, C. E. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*, 27, 379–423.

Shin, S.-i. & Choi, K.-s. (2004). Automatic Word Sense Clustering using Collocation for Sense Adaptation. In *Proceedings of the 2nd Global WordNet Conference* (pp. 1–6).

Shinyama, Y. & Sekine, S. (2006). Preemptive Information Extraction using Unrestricted Relation Discovery. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics* -, number June (pp. 304—-311). New York City, USA: Association for Computational Linguistics.

Skut, W. & Brants, T. (1998). Chunk Tagger - Statistical Recognition of Noun Phrases. *CoRR*, cmp-lg/980(1988).

Skut, W., Brants, T., Krenn, B., & Uszkoreit, H. (1998). A Linguistically Interpreted Corpus of German Newspaper Text. In *Proceedings of the 1st International Conference on Language Resources and Evaluation* Granada, Spain: European Language Resources Association (ELRA).

Srinivas, B., Doran, C., Hockey, B. A., & Joshi, A. (1996). An approach to Robust Partial Parsing and Evaluation Metrics. In *In Proceedings of the Eight European Summer School In Logic, Language and Information* (pp. 70–82).

Steinhaus, H. (1957). Sur la division des corps matériels en parties. *Bull. Acad. Polon. Sci*, 4(12), 801–804.

Stolcke, A. & Omohundro, S. M. (1994). Inducing probabilistic grammars by Bayesian Model Merging. In *Proceedings of the 2nd ICGI*.

Sumida, A. & Torisawa, K. (2008). Hacking Wikipedia for Hyponymy Relation Acquisition. In *Proceedings of the Third International Joint Conference on Natural Language Processing* (pp. 883–888).: Association for Computational Linguistics.

Surdeanu, M., Johansson, R., Meyers, A., Màrquez, L., & Nivre, J. (2008). The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. *Proceedings of the Twelfth Conference on Computational Natural Language Learning - CoNLL '08*, (August), 159.

Tan, P.-N., Steinbach, M., & Kumar, V. (2005). *Introduction to Data Mining*. Addison-Wesley.

Tesitelová, M. (1992). *Quantitative Linguistics*. Amsterdam, Netherlands; Philadelphia, USA: John Benjamins Publishing Company.

Tesnière, L. (1959). *El{é}ments de syntaxe structurale*. C. Klincksieck.

Thielen, C., Schiller, A., Teufel, S., & Stï¿½ckert, C. (1999). *Guidelines für das Tagging deutscher Textkorpora mit STTS*. Technical report, University of Stuttgart and University of Tï¿½bingen.

Thomas, W. (2010). "When nobody else dreamed of these things" - Axel Thue und die Termersetzung. *Informatik-Spektrum*, 33(5), 504–508.

Thue, A. (1914). *Probleme über Veränderungen von Zeichenreihen nach gegebenen Regeln*. Kristiania.

Toutanova, K., Klein, D., Manning, C. D., & Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1* (pp. 173—-180). Edmonton, Canada: Association for Computational Linguistics.

Toutanova, K. & Manning, C. D. (2000). Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics - Volume 13* (pp. 63—-70). Hong Kong: Association for Computational Linguistics.

Turney, P. D. (2006). Expressing Implicit Semantic Relations without Supervision. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics* (pp. 313–320).

van Dongen, S. M. (2000). *Graph clustering by flow simulation*. PhD thesis, University of Utrecht (The Netherlands).

van Zaanen, M. (2000). ABL: Alignment-based learning. In *Proceedings of the 18th International Conference on Computational Linguistics* (pp. 961—-967). Saarbrücken, Germany: Association for Computational Linguistics.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. New York City, USA: Springer-Verlag New York, Inc.

Vert, J.-P. (2002). A tree kernel to analyse phylogenetic profiles. *Bioinformatics*, 18(1), 276–284.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2), 260–269.

Wang, T., Li, Y., Bontcheva, K., Cunningham, H., & Wang, J. (2006). Automatic Extraction of Hierarchical Relations from Text. In *Proceedings of the Third European Semantic Web Conference (ESWC 2006), Budva*.

Watkins, C. (1999). Dynamic alignment kernels. In *Advances in Large Margin Classifiers*, number January (pp. 39–50). MIT Press.

Wilkinson, R. & Hingston, P. (1991). Using the cosine measure in a neural network for document retrieval. In *Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '91 (pp. 202–210). New York, NY, USA: ACM.

Wilks, Y. & Stevenson, M. (1998). The Grammar of Sense: Using Part-of-Speech Tags as a First Step in Semantic Disambiguation. *atural Language Engineering*, 4(2), 135–143.

Wilson, T., Wiebe, J., & Hoffmann, P. (2009). Recognizing Contextual Polarity: An Exploration of Features for Phrase-Level Sentiment Analysis. *Computational Linguistics*, 35(3), 399–433.

Winston, M. E., Chaffin, R., & Herrmann, D. (1987). A taxonomy of part-whole relations. *Cognitive Science*, 11(4), 417–444.

Witten, I. H. & Frank, E. (2000). *Data mining: practical machine learning tools and techniques with Java implementations*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Xia, F. & Palmer, M. (2001). Converting dependency structures to phrase structures. In *Proceedings of the first international conference on Human language technology research*, HLT '01 (pp. 1–5). Stroudsburg, PA, USA: Association for Computational Linguistics.

Yan, Y., Okazaki, N., Matsuo, Y., Yang, Z., & Ishizuka, M. (2009). Unsupervised relation extraction by mining Wikipedia texts using information from the web. *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - ACL-IJCNLP '09*, (August), 1021.

Yangarber, R., Grishman, R., Tapanainen, P., & Huttunen, S. (2000). Automatic acquisition of domain knowledge for information extraction. In *Proceedings of the 18th International Conference on Computational Linguistics*.

Zelenko, D., Aone, C., & Richardella, A. (2002). Kernel Methods for Relation Extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, number July (pp. 71—-78). Philadelphia, USA.

Zelenko, D., Aone, C., & Richardella, A. (2003). Kernel Methods for Relation Extraction. *Journal of Machine Learning Research*, 3(6), 1083–1106.

Zeng, Y., Tang, J., Garcia-Frias, J., & Gao, G. R. (2002). An Adaptive Meta-Clustering Approach: Combining the Information from Different Clustering

Results. In *Proceedings of the IEEE Computer Society Conference on Bioinformatics*, CSB '02 (pp. 276—-). Washington, DC, USA: IEEE Computer Society.

APPENDIX

A

PART-OF-SPEECH TAGSETS

A.1 PENN TREE TAGSET

Tag	Description
$	dollar
“	opening quotation mark
”	closing quotation mark
(	opening parenthesis
)	closing parenthesis
,	comma
–	dash
.	sentence terminator
:	colon or ellipsis
CC	conjunction, coordinating
CD	numeral, cardinal
DT	determiner
EX	existential there
FW	foreign word
IN	preposition or conjunction, subordinating

Tag	Description
JJ	adjective or numeral, ordinal
JJR	adjective, comparative
JJS	adjective, superlative
LS	list item marker
MD	modal auxiliary
NN	noun, common, singular or mass
NNP	noun, proper, singular
NNPS	noun, proper, plural
NNS	noun, common, plural
PDT	pre-determiner
POS	genitive marker
PRP	pronoun, personal
PRP$	pronoun, possessive
RB	adverb
RBR	adverb, comparative
RBS	adverb, superlative
RP	particle
SYM	symbol
TO	'to' as preposition or infinitive marker
UH	interjection
VB	verb, base form
VBD	verb, past tense
VBG	verb, present participle or gerund
VBN	verb, past participle
VBP	verb, present tense, not 3rd person singular

Tag	Description
VBZ	verb, present tense, 3rd person singular
WDT	WH-determiner
WP	WH-pronoun
WP$	WH-pronoun, possessive
WRB	Wh-adverb

Table 28.: Penn Tree Tagset

A.2 STUTTGART–TÜBINGEN TAGSET

Tag	Description
ADJA	attributive adjective
ADJD	adverbial or predicative adjective
ADV	adverb
APPR	preposition, left part of circumposition
APPRART	preposition with article folded in
APPO	postposition
APZR	right part of circumposition
ART	definite or indefinite article
CARD	cardinal number
FM	foreign word
ITJ	interjection
KOUI	subordinating conjunction with “zu” and infinitive
KOUS	subordinating conjunction with sentence
KON	coordinating conjunction
KOKOM	comparative conjunction

Tag	Description
NN	common noun
NE	proper noun
PDS	demonstrative pronoun that substitutes
PDAT	demonstrative pronoun that adds an attribute
PIS	indefinite pronoun that substitutes
PIAT	indefinite pronoun that adds an attribute, no article
PIDAT	indefinite pronoun that adds an attribute, with article
PPER	non-reflexive personal pronoun
PPOSS	substituting possessive pronoun
PPOSAT	attribute adding posessive pronoun
PRELS	substituting relative pronoun
PRELAT	attribute adding relative pronoun
PRF	reflexive personal pronoun
PWS	substituting interrogative pronoun
PWAT	attribute adding interrogative pronoun
PWAV	adverbial interrogative or relative pronoun
PAV	pronominal adverb
PTKZU	“zu” before infinitive
PTKNEG	negation particle
PTKVZ	particle part of separable verb
PTKANT	answer particle
PTKA	particle associated with adverb or adjective
TRUNC	first member of compound noun
VVFIN	full finite verb
VVIMP	full imperative

Tag	Description
VVINF	full infinitive
VVIZU	full infinitive with “zu”
VVPP	full past participle
VAFIN	auxilliary finite verb
VAIMP	auxilliary imperative
VAINF	auxilliary infinitive
VAPP	auxilliary past participle
VMFIN	modal finite verb
VMINF	modal infinitive
VMPP	modal past participle
XY	non word with special characters
$,	comma
$.	sentence ending punctuation
$(	other sentence signs, sentence internal

Table 29.: Stuttgart-Tübingen Tagset

B

INDUCED GRAMMAR RULES

B.1 ENGLISH GRAMMAR

Score	Context	Head	Body		
74270.63177582319	0	P#1	DT	JJ	NN
88061.77848216216	0	P#1	DT	NN	
54369.84806897123	0	P#2	TO	VB	
46920.47910752379	0	P#3	$	CD	
37468.83221512519	0	P#4	MD	VB	
19711.39645199556	0	P#5	IN	JJ	NNS
19219.0273942361	0	NNP	NNP	NNP	
17637.317918311677	0	P#1	DT	NNP	NN
18531.742363575035	0	P#5	IN	P#1	
15969.104090173903	0	P#5	IN	JJ	NN
13975.020436707462	0	P#5	IN	P#3	CD
13128.592962014069	0	P#6	IN	NNP	
10612.805772946822	0	P#5	IN	DT	NNS
9553.013073822713	0	P#5	IN	PRP$	NN
9318.700320753913	0	P#7	IN	DT	NNP

Score	Context	Head	Body		
8499.264192814693	0	NNS	JJ	NNS	
10500.654740832344	0	NN	JJ	NN	
11157.89577141539	0	P#5	IN	NN	
10841.842572036125	0	P#5	IN	NNS	
9156.91268208367	0	P#8	IN	CD	
7196.4602971582	0	P#9	NNP	POS	NN
6117.317345499176	0	P#5	IN	PRP$	NNS
6489.379111706673	0	P#1	PRP$	NN	
6046.867614410518	0	P#10	IN	DT	JJ
5852.413112390717	0	P#1	DT	NNS	

Table 30.: Excerpt from an induced English Grammar

B.2 GERMAN GRAMMAR

Score	Context	Head	Body		
85301.30911785517	0	P#1	ART	ADJA	NN
117015.77655234578	0	P#1	ART	NN	
49261.27327078488	0	NN	ADJA	NN	
29667.353413361485	0	P#2	APPRART	NN	
26686.12227771866	0	VVINF	PTKZU	VVINF	
15291.533424746702	0	P#2	APPR	P#1	
17384.356347418347	0	P#2	APPR	NN	
13754.112343966988	0	P#1	PPOSAT	NN	
11708.314692074633	0	P#2	APPR	NE	

Score	Context	Head	Body		
11077.22866653416	0	P#1	PIAT	NN	
9319.870809583754	0	P#2	APPR	CARD	NN
6310.341614290629	0	P#2	KON	NN	
6089.243463766439	0	ART	APPR	ART	
6023.08848350878	0	ART	ART	ADJA	
5070.9184982035895	0	P#3	PDAT	NN	
4600.599859217811	0	P#4	CARD	NN	
4044.144153528813	0	NE	NE	NE	
4980.356560655746	0	NE	ART	NE	
2604.3767780827943	0	NE	NE	KON	NE
2528.081738180467	0	P#1	P#1	P#2	P#2
2324.175121909172	0	P#5	APPR	CARD	
2273.6115896895276	0	VVPP	P#1	VVPP	VAFIN
2093.972408353057	0	NN	NN	P#2	
2128.1470676202857	0	P#1	APPR	P#3	
2045.8557943292808	0	P#1	P#1	KON	P#1

Table 31.: Excerpt from an induced German Grammar

SELBSTÄNDIGKEITSERKLÄRUNG

Hiermit erkläre ich, die vorliegende Dissertation selbständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialen oder erbrachten Dienstleistungen als solche gekennzeichnet.

(Ort, Datum)

(Unterschrift)

WISSENSCHAFTLICHER WERDEGANG

2002 Immatrikulation an der Universität Leipzig für das Fach Diplom Informatik mit Nebenfach Physik

2005 Vordiplom im Fach Diplom Informatik mit Nebenfach Physik

2006 Viermonatiges Praktikum an der Vanderbilt University, in Nashville, USA

2008 Abschluss Studium Diplom Informatik mit Nebenfach Physikan der Universität Leipzig

2008 – 2011 Researcher und Doktorand bei der Daimler AG, in Ulm
Schwerpunkt: Text Mining for Quality Analysis in der Abteilung Research & Development
Teilnahme an Konferenzen:

- Text Mining Services 2009 in Leipzig, Germany
- World Congress on Engineering and Computer Science 2010 in San Francisco, USA
  Best Paper Award
- 48th Annual Meeting of the Association for Computational Linguistics and Fourteenth Conference on Computational Natural Language Learning 2010 in Uppsala, Schweden
- 18th International Nordic Conference of Computational Linguistics 2011 in Riga, Lettland
- 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies 2011 in Portland, USA

SEIT 2011 Researcher bei der ExB Research & Development GmbH
Schwerpunkt: Unsupervised Natural Language Processing