Learning to Identify Antonymy

Lernen, wie man Gegenteile findet

Master-Thesis von Peter Klöckner

Tag der Einreichung:

1. Gutachten: Prof. Dr. Chris Biemann
2. Gutachten: Dr. Martin Riedl

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Informatik
Language Technology Group

Learning to Identify Antonymy
Lernen, wie man Gegenteile findet

Vorgelegte Master-Thesis von Peter Klöckner

1. Gutachten: Prof. Dr. Chris Biemann
2. Gutachten: Dr. Martin Riedl

Tag der Einreichung:

Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 2. August 2016

(Peter Klöckner)

Abstract

Detecting whether two words have an opposite meaning and thus are antonyms is relevant for many applications and methods in Natural Language Processing such as Lexical Substitution or Information Retrieval.

In this thesis, we present a supervised approach to antonymy identification that relies on small lists of known antonyms and large corpora. We train part-of-speech category specific classifiers using a diverse set of features. The features exploit clues for antonymy residing in different layers of language. The most prominent features make use of syntacmatic relations of antonym pairs. Other features capture morphological clues, co-occurrence probabilities and distributional similarity.

We evaluate our classifiers intrinsically on a data set we compiled from WordNet containing word pairs of different semantic relations such as antonymy, synonymy and hyponymy and on antonymy versus synonymy evaluation sets. Apart from that, we evaluate our classifiers extrinsically on “most contrasting word” questions.

On the one hand, our evaluations show that pattern based information can identify antonyms precisely. On the other hand, the evaluations reveal that the recall is not too high, even though we use large corpora.

Zusammenfassung

Antonymy bzw. Gegenteiligkeit zu erkennen ist von großer Relevanz für einige Disziplinen in der Sprachtechnologie wie der lexikalischen Substitution und der Informationsrückgewinnung.

In dieser Arbeit wird ein überwachter Ansatz für das Erkennen von Antonymen präsentiert, der auf kleinen Listen von Beispiel-Antonymen und großen Korpora basiert. Es werden Klassifikatoren für die Wortarten Nomen, Verben und Adjektive trainiert. Verschiedene Merkmale werden herangezogen, um Antonymy zu erkennen. Darunter befinden sich morphologische Hinweise, Hinweise aus Kookkurrenzstatistiken, die distributionelle Ähnlichkeit der Wörter und besonders Informationen zu den syntaktischen Mustern, in denen die zu klassifizierenden Wortpaare vorkommen.

Die Klassifikatoren werden intrinsisch auf einem mittels WordNet erstellten Datensatz und weiteren Datensätzen aus der Literatur evaluiert. Der WordNet-Datensatz enthält Worpaare verschiedenster semantischer Relationen, Antonyme, Synonyme, Hyperonym/Hyponym- und Meronym-Holonym-Paare, und Wortpaare, die keine Relation auweisen. Die Datensätze aus der Literatur enthalten nur Antonym- und Synonym-Paare. Des weiteren werden die Klassifikatoren extrinisisch auf einem Antwort-Auswahl-Test evaluiert. Bei diesem Test geht es darum zu einem Zielwort aus den Wörtern, die als Antwortmöglichkeiten bestehen, das Wort herauszusuchen, das am stärksten im Kontast zu dem Zielwort steht.

Die Evaluation zeigt, dass mit den sich vorwiegend auf syntaktische Muster verlassenden Klassifikatoren Antonyme zwar präzise bestimmt werden, aber viele Antonym-Paare nicht als solche erkannt werden.

Acknowledgment

I would like to thank my thesis advisor Dr. Martin Riedl for taking the time for regular in-person meetings and thoroughly answering all my questions. I thank Prof. Dr. Chris Biemann for his guidance. Thanks to Seid, Steffen, Eugen and the other members of the Language Technology group for providing me with their ideas for my topic.

Lastly, thanks to my family, roommates and friends for showing support all the way.

Contents

Acronyms

DT Distributional Thesaurus. 16–19, 33, 57, 65

GRE Graduate Record Examination. 8, 29, 58–61, 67

k-NN k-nearest neighbor classifier. 21

LLR Log-likelihood Ratio. 14, 15

LMI Lexicographers Mutual Information. 14, 15, 17

NLP Natural Language Processing. 8, 12, 13, 21

PMI Pointwise Mutual Information. 13, 14, 17

SVM Support Vector Machine. 24, 50

1 Introduction

Natural Language Processing (NLP) is a field of computer science concerned with the automatic processing of natural (human) language. One challenge in NLP is to get the computer to understand text, to gather structured knowledge from text that can be reasoned with.

To solve this task several subtasks have to be solved. The computer has to be able to identify entities in texts, for instance it has to understand that „FC Bayern Munich“ is one entity. Furthermore, the computer needs to be able to determine which sense of a word is used in a sentence in case a word has more than one meaning. E.g. the computer needs to understand if the word „football“ in a sentence refers to American football or soccer.

Then, the computer needs to be able to extract the semantic relations between word senses from text. There are many different semantic relations. In this thesis, however, we concentrate on the automatic identification of the relation antonymy. A word pair is antonymous if it is opposite in meaning, e.g. „good bad“ is an antonym pair.

This first chapter motivates the importance of identifying antonymy, describes the research goal of this thesis and outlines the thesis structure.

1.1 Motivation

Identifying antonymy/opposition is important for many applications of NLP. For instance, in the case of Lexical Substitution, the task of choosing a suitable replacement for a word in a sentence, filtering out antonymous candidate words is essential. Other applications which can profit from antonymy identification include Machine Translation [Roth and Schulte im Walde, 2014], Sentiment Analysis [Mohammad et al., 2013] and Information Retrieval [Lin et al., 2003].

There are manually created taxonomies which contain information about antonymous relations. One of the prominent examples for the English language is WordNet [Miller, 1995]. For the German language there is for example GermaNet [Hamp and Feldweg, 1997]. Both, WordNet and GermaNet, encode antonymous relations for adjectives, nouns and verbs.

Although manually created taxonomies are often used to support the initially mentioned NLP tasks, such taxonomies have several disadvantages. First, their coverage is limited. For instance, according to Mohammad et al. [2013] more than 90% of the answer pairs in the Graduate Record Examination (GRE) closest-to-opposite questions cannot be found in WordNet as antonym pairs. One reason for low coverage is that language changes rapidly. Second, manually creating such taxonomies is a slow and labor-intensive process [Snow et al., 2005] which requires the work of experts and is thus also expensive. Apart from that, for many languages, especially minority languages, manually created taxonomies simply do not exist.

One way to assign meaning to words and to measure word similarity without depending on manually created lexical resources is to rely on distributional semantics. According to the *distibutional hypothesis* [Harris, 1954] words that occur in the same contexts tend to have similar meanings.

However, not only synonyms or near-synonyms but also hyponyms, cohyponyms, meronyms, antonyms and otherwise related words can be found among distributionally similar words. Thus, only with information about the distributional similarity of a word pair the type of semantic relation can not be uncovered which directly leads us to our research goal described in the next section.

1.2 Research Goal

In this thesis, we want to examine how antonymy can be identified and also discriminated from other lexical relations, such as hyponymy or synonymy, using a supervised approach depending on different feature categories including word representations, morphological features, co-occurrence frequencies and most prominently pattern-based features.

The final goal is to develop a system which can identify antonymous pairs of words among distributionally similar words. The system should work for English adjective, noun and verb pairs.

1.3 Thesis Structure

In the following, we briefly outline the thesis structure. The basic knowledge necessary to understand the research conducted in this thesis is provided in Chapter 2. This chapter describes some theoretical views on antonymy and gives an introduction to distributional semantics and machine learning. Thereafter, we discuss previous research on the task of antonymy identification in Chapter 3. We present our own methods in Chapter 4 which we evaluate in Chapter 5. The thesis is concluded in Chapter 6 which also gives a short outlook over possible future work.

2 Background

This chapter provides an introduction to technical terms, concepts and algorithms, whose understanding is necessary to understand the core research on antonymy identification described in this thesis. The chapter starts with a short introduction to the semantic relation antonymy, which is followed by a broad overview of general linguistic terminology. In this thesis, lexical databases are exploited to garner data for training and thus are introduced in another section. The next sections present the idea of distributional semantics and a framework for determining distributional similarity and describe technical foundations of the computation of distributional similarity. The chapter ends with a primer on supervised machine learning and classification, the learning technique used to grasp the concept of antonymy.

2.1 What is antonymy?

There exist different theoretical definitions on what antonymy is and whether antonymy and opposition are the same or antonymy is a subcategory of opposition. Cruse [1986] defines antonymy as a relation between two gradable adjectives like “cold” and “hot”. The adjectives are supposed to point into the polar directions of a scale and it should be possible to modify them with words such as very and slightly. Murphy [2003] thinks of antonyms and opposites as synonymous terms. Apart from gradable adjectives there are also non-gradable binary opposites like “dead” and “alive” and furthermore triplets or even more words that stand in opposition with each other like “to try”, “to succeed”, “to fail” and the four seasons “winter”, “spring”, “summer”, “autumn” [Lobanova, 2012]. The difficulty to define antonymy make [Lobanova, 2012] claim, that the “automatic extraction of opposites is a much more difficult task than the identification of such relations as hyponymy or synonymy”. In this thesis, we depend on the assessment of annotators of the used lexical resources on whether a word pair is an antonymous one. Considering some example pairs from WordNet it seems that a rather broad definition of antonymy was used constructing WordNet, since pairs of gradable, non-gradable and pairs of antonymous nouns, verbs and adjectives can each be found in this lexical resource.

2.2 Linguistic Terminology

The field of linguistics can be divided into several subareas each addressing one level of language [Jurafsky and Martin, 2000]. The process of language understanding can then be seen as a bottom-up task iterating through the different levels or layers of language. The lowest layer, the level of sounds, consists of phonetics and phonology, whereby the former deals with all possible human sounds and the later with the sounds, phonemes, used to form human language. The next layer is concerned with the smallest meaning-bearing units of human language, the morphemes, and thus is call morphology. Morphemes can among others be prefixes like “un-” or “non-”, which make up the beginning of a word, or suffixes like a plural “-s” at the end of a word. Words are in fact sequences of morphemes. The syntax layer resides above the morphology layer. Syntax is about the structure of sentences, which are made of words (, which themselves are composed of morphemes, which are in turn sequences or phonemes). The next level, semantics, is about the meaning of words, phrases, sentences and documents. The top-most layer, pragmatics, covers the purpose and intent of spoken or written language in context. Table 2.1 provides an overview of the different levels of language.

Level of Language	**Short Description**
Pragmatics	Purpose and intent of language use
Semantics	Meaning of language
Syntax	Structure of language
Morphology	Smallest meaning-bearing units
Phonetics/Phonology	Sounds/Basic sounds that form language

Table 2.1: Levels of language

Computational understanding of language can follow the theory of language layers by using a processing pipeline whose consecutive steps each rely on the output of the respective previous step. A common first step in analyzing natural language documents is to split the text into sentences, sentence splitting, and to identify tokens, tokenization. Tokens are words and other language elements like commas, points and exclamation marks. A subsequent step is lemmatization, the process of assigning basic word forms called lemmas to the inflected word forms occurring in the raw text. To reach the syntax level part-of-speech tags have to be assigned to the words and the sentences have to be parsed. A list of the part-of-speech tags used in this thesis can be found in the appendix in Table 7.1. Parsing of a sentence can be done with a dependency parser. Such a parser produces a dependency tree. The tree contains the syntactic relations between words. An example of a dependency tree is shown in Figure 2.3.

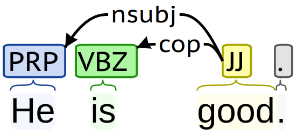


Figure 2.1: Dependency parse from Stanford CoreNLP[1]

The nominal subject of the sentence „he“, a personal pronoun is linked to the adjective „good“, which is linked to the verb „is“. The relation types used in this thesis are described in De Marneffe and Manning [2008].

A crucial part of semantics is to understand semantic relations between word senses. Apart from antonymy whose basics were covered in section 2.1, other semantic relations include hyponymy/hypernymy, co-hyponymy, holonymy/meronymy and synonymy. Hyponyms are superordinate terms of hypernyms, subordinate terms. An example for a hypernym/hyponym pair is tree and oak, as every oak is a tree. Co-hyponyms share a common hypernym. An example for co-hyponyms are oak and chestnut, as both are trees. A meronym is a part of a holonym, e.g. a branch is a part of a tree. Note that semantic relations are defined between word senses and not between words. E.g. an oak is not a hyponym of the data structure tree but of trees in a botanical sense.

2.3 Lexical Databases

Lexical databases are machine-readable lexical resources. Lexical resources are in turn composed of one or more dictionaries mapping words to useful information such as different senses and their definitions, pronunciation, translation and related words. Information about related words is of particular interest in this thesis. We, in fact, compile the training set for our antonym classifiers with the help of WordNet. Hence, we shortly introduce this lexical database at this point.

WordNet

A large English language lexical database is WordNet. As of version 3.0, WordNet contains more than 150000 unique words. Words are either nouns, verbs, adjectives or adverbs, meaning there is e.g. no information about pronouns, prepositions or determiners.

WordNet groups words in so called synsets, which consist of words expressing one sense. All of the words in one synset belong to the same part-of-speech category. A synset has a short definition ("gloss") and may be accompanied with one or more phrases or sentences exemplifying the usage. Synsets are linked among each other by semantic relations.

Synsets consisting of nouns are linked by hyponymy and meronymy. Synsets of verbs are linked by troponymy. Adjectival synsets are ordered by antonymy.

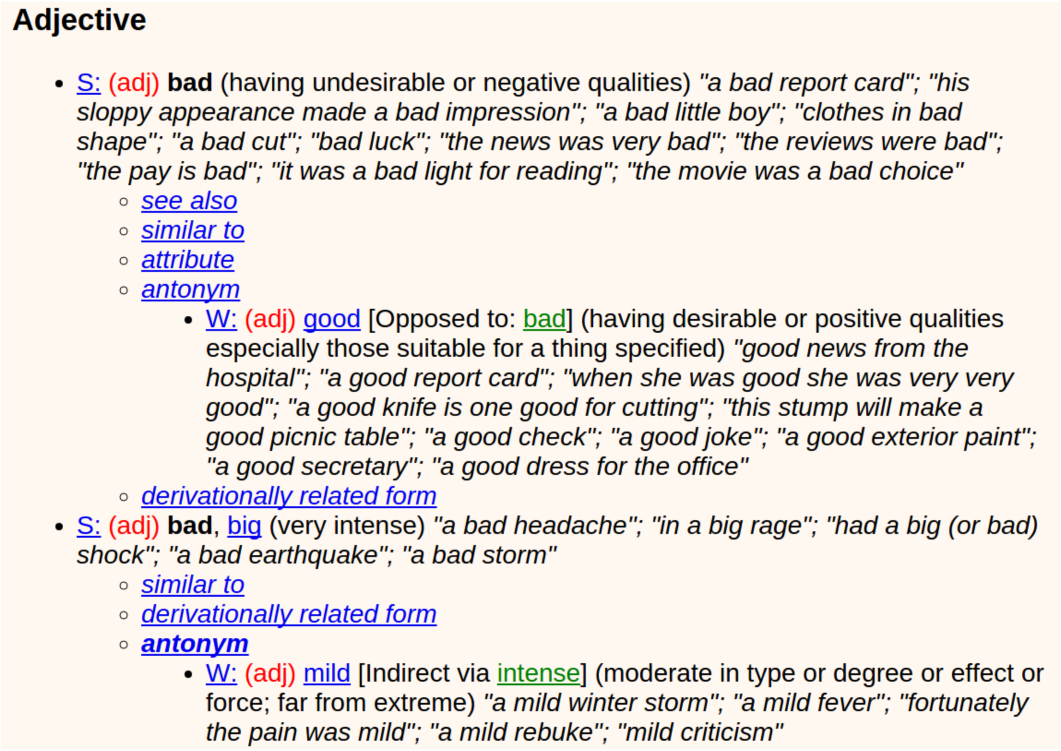
Semantic relations are mostly encoded between synsets of the same part-of-speech category. An exception to this are for instance „morphosemantic“ links connecting words sharing a stem, e.g. act and actor.

To illustrate WordNet Figure 2.2 shows two of the adjectival synsets that contain the word „bad“. The first synset is solely made of the word „bad“ and is defined by the gloss „having undesirable or negative qualities“. This synset is linked to the antonymous synset „good“. Another synset of the word „bad“ is described by „very intense“ and is linked to the synset „mild“.

2.4 Association Measures

An association measure computes an association score for a pair of items. A high association score commonly indicates strong association and thus that a pair of items tends to co-occur. In the context of NLP, the pair might consist of language elements such as words, N-grams or patterns. Distributional Semantics, which is presented in Section 2.5, relies on association scores of pairs of language elements. Furthermore, the identification of semantically related words is often based on pair-pattern association scores. Hence, in this section we shortly discuss the computation of association scores and introduce often used association measures.

[2] `http://wordnetweb.princeton.edu/perl/webwn` (last accessed: 2016/03/01)



Adjective

- S: (adj) **bad** (having undesirable or negative qualities) *"a bad report card"; "his sloppy appearance made a bad impression"; "a bad little boy"; "clothes in bad shape"; "a bad cut"; "bad luck"; "the news was very bad"; "the reviews were bad"; "the pay is bad"; "it was a bad light for reading"; "the movie was a bad choice"*
    - *see also*
    - *similar to*
    - *attribute*
    - *antonym*
        - W: (adj) good [Opposed to: bad] (having desirable or positive qualities especially those suitable for a thing specified) *"good news from the hospital"; "a good report card"; "when she was good she was very very good"; "a good knife is one good for cutting"; "this stump will make a good picnic table"; "a good check"; "a good joke"; "a good exterior paint"; "a good secretary"; "a good dress for the office"*
    - *derivationally related form*
- S: (adj) **bad**, big (very intense) *"a bad headache"; "in a big rage"; "had a big (or bad) shock"; "a bad earthquake"; "a bad storm"*
    - *similar to*
    - *derivationally related form*
    - ***antonym***
        - W: (adj) mild [Indirect via intense] (moderate in type or degree or effect or force; far from extreme) *"a mild winter storm"; "a mild fever"; "fortunately the pain was mild"; "a mild rebuke"; "mild criticism"*

Figure 2.2: Extract of the adjectival synsets of „bad" from the online WordNet interface[2]

The computation of association scores is based on the contingency table of an item pair [Evert, 2005, pp. 75]. Four important scores can be found in a contingency table (or computed from information in a contingency table). First, the joint frequency, or co-occurrence frequency, of the pair $n_{A,B}$, denoting how often the pair's items co-occur. Second and third the component frequencies, or marginal frequencies, of a pair, thus, the single frequencies n_A and n_B of the pair's items. And fourth and last the total number of pairs N.

Several different association measures exist, which differ for instance in the assumptions made about the distribution of the data, the computation speed and the ease of interpretation. In the following, we introduce three association measure, which have proven to work well for NLP tasks.

Pointwise Mutual Information

Pointwise Mutual Information (PMI) [Church and Hanks, 1990] compares the joint probability of two events x and y with the probabilities of x and y (the marginal frequencies).

$$PMI(x;y) = \log \frac{p(x,y)}{p(x)p(y)} \tag{2.1}$$

Assuming x and y being independent, the joint probability should equal the product of the two events' probabilities. Hence, taking the logarithm into account, the PMI of two events is 0 if the events are statistically independent. If two events occur more often than expected, their PMI-score is positive; if the events occur less often than expected, the PMI-score is negative. Note that PMI is a symmetric measure.

Since the real joint probability and the real marginal probabilities often are unknown, the joint probability is normally approximated with $\frac{n_{A,B}}{N}$ and the marginal probabilities with $\frac{n_A}{N}$ and $\frac{n_B}{N}$, yielding:

$$PMI(A,B) = \log \frac{N n_{A,B}}{n_A n_B} \tag{2.2}$$

A high PMI-score does not necessarily imply a truly high association of two terms. Assuming two words occur statistically perfectly dependent, meaning the two words always occur together, the following holds:

$$PMI(x;y) = \log \frac{p(x,y)}{p(x)p(y)} = \log \frac{p(x|y)p(y)}{p(x)p(y)} = \log \frac{p(x|y)}{p(x)} = \log \frac{1}{p(x)} = \log \frac{1}{p(y)} \tag{2.3}$$

Under this assumption thus the following applies: The more infrequent the words are, the higher is the PMI-score.

Apart from that it is more likely that infrequent words coincidentally co-occur relatively often than frequent words. Hence, PMI is biased towards infrequent words.

This weakness of PMI may be addressed by using frequency thresholds or resorting to the following measure:

Lexicographers Mutual Information

Lexicographers Mutual Information (LMI) [Bordag, 2008] weights the PMI-score with the joint frequency of the events:

$$LMI(A,B) = n_{A,B} \cdot PMI(A,B) \tag{2.4}$$

The weight factor effectively reduces the bias of PMI towards infrequent words. PMI and LMI are measures from information theory; in contrast to that, the measure introduced next is based on hypothesis testing:

Log-likelihood Ratio

Log-likelihood Ratio (LLR) [Dunning, 1993] examines two hypotheses potentially explaining the frequency $n_{A,B}$:

- Hypothesis H_0: $P(A|B) = p = P(B|A)$
- Hypothesis H_1: $P(A|B) = p_1 \neq p_2 = P(B|\neg A)$

According to hypothesis H_0 the pair's items occur independently. Hypothesis H_1 states that the items occur dependently. Assuming an underlying binomial distribution and using maximum likelihood LLR then gives an association score denoting whether hypothesis H_0 or H_1 is more likely:

$$
\begin{aligned}
LLR(A,B) &= 2\log\frac{H_0}{H_1} \\
&= -2\cdot\log\lambda \\
&= 2\cdot[N\log n - n_A\log n_A - n_B log n_B + n_{A,B}\log n_{A,B} \\
&\qquad + (N - n_A - n_B + n_{A,B})\log(N - n_A - n_B + n_{A,B} \\
&\qquad + (n_A - n_{A,B})\log(n_A - n_{A,B}) \\
&\qquad + (n_B - n_{A,B})\log(n_B - n_{A,B}) \\
&\qquad - (N - n_A)\log(N - n_A) - (N - n_B)\log(N - n_B)]
\end{aligned}
\tag{2.5}
$$

As can be seen by studying their respective formulas, LMI is faster to compute than LLR, which makes LMI a good choice if large amounts of data have to be analyzed.

2.5 Distributional Semantics

> You shall know a word by the company it keeps
>
> [Firth, 1957]

De Saussure [[1916] 1983] differentiates between two viewpoints we make use of when reading and trying to understand text. The syntagmatic and the associative viewpoint. The syntagmatic viewpoint is about the linear ordering of textual units, about grammatical dependencies. The associative viewpoint, also called paradigmatic viewpoint, is concerned with as de Saussure [[1916] 1983] states "terms in absentia", associated terms residing in the readers memory.

I	play	soccer
you	do	karate
they	go	swimming

I	play	soccer
you	do	karate
they	go	swimming

Table 2.2: Syntagmatic and paradigmatic relations (from left)

Years after the foundational work of de Saussure [[1916] 1983] the *distributional hypotheses* was formulated [Harris, 1954]:

> *The distribution of an element is the total of all environments in which it occurs, i.e. the sum of all the (different) positions (or occurrences) of an element relative to the occurrence of other elements.*

Instead of using the abstract term „environment“ Miller and Charles [1991] later directly focused on words and introduced the notion of semantic similarity in their *strong contextual hypothesis*:

> *Two words are semantically similar to the extent that their contextual representations are similar.*

Miller and Charles [1991] showed that semantic similarity as defined in their *strong contextual hypothesis* correlates highly with human assessments on word similarity. This makes distributional similarity (the similarity of contexts) a good tool to approximate paradigmatic relations, as defined by de Saussure [[1916] 1983], between words.

The emergence of large corpora and the increase of computation power made it possible to compute distributional similarity between all the word pairs in a huge vocabulary and store the results in a Distributional Thesaurus (DT).

Examples for distributional thesauri include [Lin, 1998], [Curran, 2002] and [Biemann and Riedl, 2013].

In this thesis, we work with DTs generated with the standard parameters of the JoBimText framework [Biemann and Riedl, 2013]. The details of this framework are described next.

The JoBimText Framework

The JoBimText framework implements a DT computation pipeline using Hadoop's MapReduce[3] paradigm. Hadoop makes it possible to process large amounts of (textual) data in reasonable time by distributing the computation to several servers and/or threads, which run in parallel.

The JoBimText framework's name stems from the terminology used to denote the key language elements `Jos` and their context features `Bims`. `Jos` and `Bims` may be (and often are) words, but are not restricted to being words. The framework's pipeline to compute the DT relies on a list of tuples $< x, y >$ where x is a `Jo` and y is a `Bim`. This list of tuples is created via a so called holing operation on the text input of the system. The holing operation defines the data type of the `Jos` and `Bims` and operates on most commonly preprocessed text input. Lets elaborate on the concept of key language elements, context features and the holing operation on this example sentence:

Sentence:	**Aubameyang**	**can**	**not**	**stop**	**scoring**	**.**
Position:	**1**	**2**	**3**	**4**	**5**	**6**

For our first example we define that `Jos` are words. Furthermore, we let our holing operation work on a one word window to the right of each word to extract the context features `Bims`. We assume that we process tokenized textual input. These definitions produce the following list of tuples:

$$< \$_0,(@;Aubameyang_1) >, < Aubameyang,(@;can_2) >, < can,(@;not_3) >,$$
$$< not,(@;stop_4) >, < stop,(@;scoring_5) >, < scoring;(@;._6) >, < .,(@;\$_7) >$$

Note that @ is the hole symbol. It indicates the position of the `Jo`.

For our second more advanced example we assume that in a preprocessing step of our system the text input was parsed with a dependency parser. Again, as keys we use words, but our holing operation

[3] `http://hadoop.apache.org`

Parameter	Description
t	minimum number of term-feature occurrences
s	minimum significance of a context feature
p	maximum number of features per language element

Table 2.3: Pruning Parameters

now works on the generated dependency trees. We assume the dependency tree for our example sentence looks like Figure 2.3:

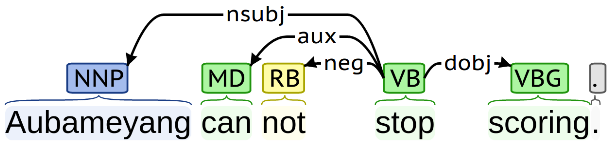


Figure 2.3: Dependency parse from Stanford CoreNLP[4]

Now this would be the list of generated $<\texttt{Jo},\texttt{Bim}>$ tuples:

$< Aubameyang, (nsubj; stop_4; @) >, < can, (aux; stop_4, @) >, < not_3, (neg; stop_4, @) >,$
$< stop_4, (neg; @; not_3) >, < stop_4; (aux; @; can_1) >, < stop_4, (nsubj; @; Aubameyang_1) >,$
$< stop_4; (dobj; @; scoring_5) >, < scoring_5; (dobj; stop_4; @) >$

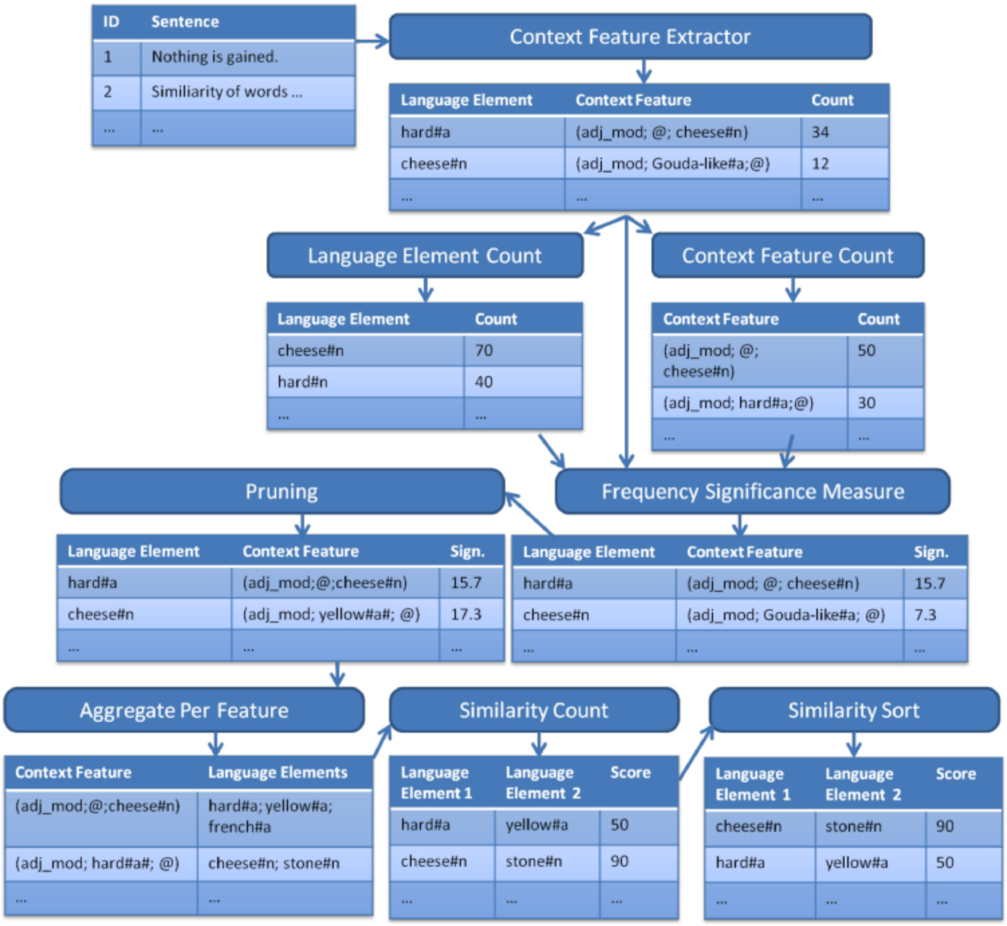
Using dependency trees for context feature generation instead of only words in a small window might make a lot of sense. For instance, in the above sentence „scoring“ describes the scorer Aubameyang quite good and is only two dependency tree hops distant of Aubameyang. But even a three word window would not yield „scoring“ as a context feature of Aubameyang.

Generating the list of language element and context feature tuples and counting how often a distinct tuple occurs in the training data is the first step of the DT computation pipeline in the JoBimText Framework. This step is done in the *Context Feature Extractor*. The full computation pipeline is shown in Figure 2.4.

Once the *Context Feature Extractor* terminates, the frequencies of the occurring language elements (*Language Element Count*) and the context features (*Feature Count*) are summed up, since these counts are needed for the computation of the significance of term-feature pairs with measures implemented in *Frequency Significance Measure*; for instance LMI or PMI (see section 2.4).

Next, the generated data is pruned (*Pruning*). Three parameters are used for pruning. If a term-feature pair occurs less than t times or is less significant than s, the pair is discarded. Also, for every term only the p most significant features are kept. See Table 2.3 for a summary of the pruning parameters.

Pruning is an important step to make the computation scalable. According to [Biemann and Riedl, 2013], the whole DT, using 120 million sentences as corpus, can be computed on a small Hadoop cluster (64 cores on 8 servers) in „well under a day“. Apart from that pruning makes sense semantically. Insignificant term-feature pairs should not contribute to the paradigmatic similarity of terms.



ID	Sentence
1	Nothing is gained.
2	Similiarity of words ...
...	...

Language Element	Context Feature	Count
hard#a	(adj_mod; @; cheese#n)	34
cheese#n	(adj_mod; Gouda-like#a;@)	12
...	...	...

Language Element	Count
cheese#n	70
hard#n	40
...	...

Context Feature	Count
(adj_mod; @; cheese#n)	50
(adj_mod; hard#a;@)	30
...	...

Language Element	Context Feature	Sign.
hard#a	(adj_mod;@;cheese#n)	15.7
cheese#n	(adj_mod; yellow#a#; @)	17.3
...	...	...

Language Element	Context Feature	Sign.
hard#a	(adj_mod; @; cheese#n)	15.7
cheese#n	(adj_mod; Gouda-like#a; @)	7.3
...	...	...

Context Feature	Language Elements
(adj_mod;@;cheese#n)	hard#a; yellow#a; french#a
(adj_mod; hard#a#; @)	cheese#n; stone#n
...	...

Language Element 1	Language Element 2	Score
hard#a	yellow#a	50
cheese#n	stone#n	90
...	...	...

Language Element 1	Language Element 2	Score
cheese#n	stone#n	90
hard#a	yellow#a	50
...	...	...

Figure 2.4: The DT computation workflow in the JoBimText framework; image from Biemann and Riedl [2013]

The final steps of the DT computation include aggregating the terms by feature (*Aggregate per feature*) and sorting the list of term-feature pairs by significance.

The resulting list makes it possible to calculate a second-order (paradigmatic) similarity between language elements using Equation 2.6.

$$
\begin{aligned}
sim(t_1,t_2) = \sum f \in rankedfeatures(t_1,p) \cap rankedfeatures(t_2,p) \\
\wedge (f(t_1) > t \wedge f(t_2) > t) \wedge (score(f) > s \wedge score(f) > s)1
\end{aligned}
\tag{2.6}
$$

Table 2.4 provides the function definitions used in the above similarity function:

Function	**Description**
$f(t)$	frequency of a language element
$score(f)$	significance score of a context feature
$rankedfeatures(f,p)$	return the p most significant features

Table 2.4: Function definitions for the similarity function

Thereby, p plays an important role. Infrequent terms naturally have on average less features than frequent terms. Thus, the definition of distributional similarity as stated above is biased towards more frequent terms. Consider an infrequent term having only 500 features and p being 1000. Now, this infrequent term can at best reach a similarity score of 500 with any other term. Hence, it is important not to choose a large number for p. On the other hand, p should of course also not be too small.

2.6 Classification

Machine Learning can be divided into several fields including supervised learning and unsupervised learning. Unsupervised learning is about finding structures and patterns in data. A common task in unsupervised learning is to divide data into coherent clusters/classes. In contrast to unsupervised machine learning in supervised learning class labels for the data at hand are available. Then, the task is to come up with general rules on what defines the present classes. These general rules can be remembered in a model, which a classifier can use to assign class labels to new data points. For example in our thesis, we want to learn what defines antonymy from a set of data points belonging to each one pair of words and information about whether the pair consists of antonyms or not. In the following, classification is introduced in a more thorough and formal way.

Classification is the task of assigning a class label y to a feature vector $\mathbf{x} = (x_1, x_2, x_3, ..., x_k)$ where x_f is called a feature value in the range $vals(f)$ of the feature f and k is the number of features. Classifiers are trained on a training set T consisting of m labeled instances of the form $(\mathbf{x}^{(i)}, y^{(i)}) = (x_1^{(i)}, x_2^{(i)}, x_3^{(i)}, ..., x_k^{(i)}, y^{(i)})$ with $i = 1, \ldots, m$.

There are plenty of different classification algorithms. According to Domingos [2012] learning algorithms differ along three dimensions:

- the representation used, which is called the *hypothesis space* of the classifier
- the evaluation function used (also called *objective function* or *scoring function*)
- the optimization technique, which is used to find the best classifier

Table 2.5 shows some *hypotheses spaces* together with example algorithms that use the type of representation.

Representation	**Example algorithms**
Instances	K-nearest neighbors, Support vector machines
Hyperplanes	Naive Bayes, Logistic regression
Decision trees	Decision trees, Random forests

Table 2.5: Common hypotheses spaces of classifiers

In the following, we give a short introduction to each of the example algorithms in Table 2.5.

2.6.1 Naive Bayes

Naive Bayes methods are a set of classifiers that use the Bayes' theorem to make predictions:

$$P(y, x_1, ..., x_n) = \frac{P(y)P(x_1, ..., x_n|y)}{P(x_1, ..., x_n)} \tag{2.7}$$

The classifiers are called „naive“, because they make the assumption that every pair of features is statistically independent:

$$P(x_i|y, x_1, ..., x_{i-1}, x_{i+1}, ..., x_n) = P(x_i|y) \tag{2.8}$$

Using the Bayes' theorem and the „naive“ assumption, the probability of a class y given a feature vector x can then be calculated in the following way:

$$P(y, x_1, ..., x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, ..., x_n)} \tag{2.9}$$

Since $P(y, x_1, ..., x_n)$ is constant for a feature vector and any class, the class $\hat{y}$ with the highest probability is chosen as prediction:

$$\hat{y} = \arg\max_x P(y)\prod_{i=1}^{n} P(x_i|y) \tag{2.10}$$

Different naive Bayes classifiers then differ by how the distribution of $P(x_i|y)$ is modeled. Despite the „naive“ assumption naive Bayes classifiers can be very good at classification tasks in NLP, for example text categorization [Rennie et al., 2003].

2.6.2 K-Nearest Neighbors

The k-nearest neighbor classifier (k-NN) is one of the simplest approaches of instance-based learning. The algorithm does not rely on a complex training, instead it just memorizes the training examples. For the classification of a feature vector the algorithm determines the k nearest training examples to the feature vector. Then, the new sample gets labeled with the class that the majority of the nearest k training examples are labeled with. Figure 2.5 illustrates the classification procedure.

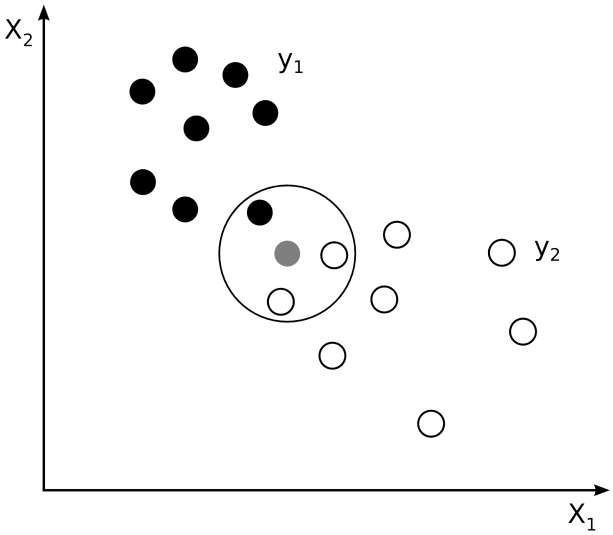


Figure 2.5: Example of a classification with a 3-nearest neighbors classifier. The new sample (in grey) is assigned to class y_2.

Two parameters are of importance for k-NN: the value k and the used distance function. A too small value of k makes the classifier vulnerable to noise in the training examples, a large value of k may lead to too many other (distant) samples influencing the prediction. Common distance function are the *Manhattan distance*

$$manhattan(a,b) = \sum_{i=1}^{k} |a_i - b_i| \tag{2.11}$$

, the *Euclidean distance*

$$euclidean(a,b) = \sqrt{\sum_{i=1}^{k} (a_i - b_i)^2} \tag{2.12}$$

and the *cosine similarity*.

$$cosine(a,b) = \frac{\sum_{i=1}^{k} a_i b_i}{\sqrt{\sum_{i=1}^{k} a_i}\sqrt{\sum_{i=1}^{k} b_i}} \tag{2.13}$$

2.6.3 Logistic Regression

Although the name suggests otherwise, logistic regression is a statistical model, which can be used for classification. Logistic regression is a *generalized linear model*, meaning it solves a problem of the form:

$$h_\theta(x) = f(\theta^\top x + \theta_0) \tag{2.14}$$

Thereby, $f(.)$ is called an *activation function*. In the case of logistic regression the activation function is the *logistic sigmoid* function $\sigma(a) = \frac{1}{1+\exp(-a)}$.

The parameters/weights θ can be determined using maximum likelihood.

Assuming that

$$P(y=1|x;\theta) = h_\theta(x) P(y=0|x;\theta) = 1 - h_\theta(x) \tag{2.15}$$

or more compactly written

$$p(y|x;\theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y} \tag{2.16}$$

the likelihood of the parameters is:

$$L(\theta) = \prod_{i=1}^{m} (h_\theta(x^{(i)}))^{y^{(i)}} (1 - h_\theta(x^{(i)})^{1-y^{(i)}} \tag{2.17}$$

Thus, the log likelihood is:

$$l(\theta) = \sum_{i=1}^{m} y^{(i)} \cdot \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)})) \tag{2.18}$$

The derivative of the log likelihood function is:

$$\frac{\delta}{\delta\theta_j} l(\theta) = (y - h_\theta(x)) x_j \tag{2.19}$$

Having the derivative of the log likelihood function, the parameters maximizing this function can for instance be found with the gradient ascent algorithm or more advanced algorithms for solving a convex optimization problem.

Logistic regression is a standard linear classifier, which does not need long for training and classification even if there are lots of features.

2.6.4 Decision Trees

There are two main types of decision trees in Machine Learning. First, classification trees and second regression trees. Since this is a chapter about classification, we will focus on the former ones. Classification trees define how a class label is predicted based on the feature values of an instance. The nodes of classification trees consist of decision rules based on one of the features. The leaves of the tree represent the class labels, which will be predicted if the classification process of an instance ends in the particular leaf. A simple example of a classification tree is shown in Figure 2.6.

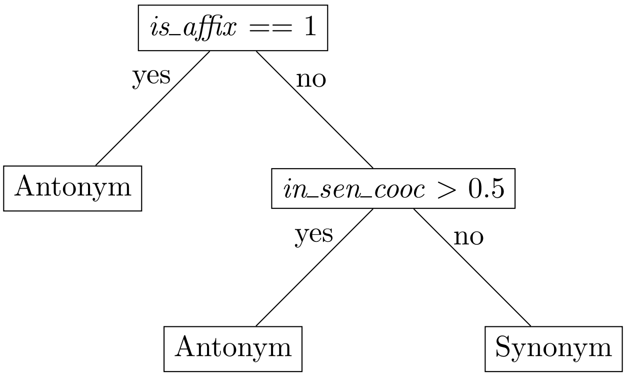


Figure 2.6: Example of a classification tree

Based on the two features *is_affix* and *in_sen_cooc* word pairs are classified as either synonyms or antonyms. *is_affix* has the feature value 1 if one of the words can be formed by concatenating the other word with an affix indicating antonymy and 0 if not. *in_sen_cooc* is the normalized probability of the two words occuring together in one sentence. According to our simple toy example, a word pair is classified as being antonymous if *is_affix* is 1 or if the words are very likely to co-occur in one sentence.

Algorithms for classification tree learning usually recursively construct the trees top down. In every step the feature is chosen that best splits the training examples. Therefore, a metric is used that defines how much is gained by splitting on a feature. Then, a new node is created containing the decision rule on the feature and the steps are repeated for the new subsets until there are only leaves that contain instances all having the same class. One popular metric for choosing on which feature to split is the Information Gain (IG), which is e.g. used in the decision tree learning algorithm C4.5 [Quinlan, 1993].

$$IG(T,f) = H(T) - \sum_{v \in vals(f)} \frac{|\{x \in T | x_f = v\}|}{|T|} \cdot H(\{x \in T | x_f = v\}) \tag{2.20}$$

IG relies on the Entropy H, which is a measure of impurity.

$$H(X) = -\sum_{i=1}^{n} p(x_i) \cdot \lg(p(x_i)) \tag{2.21}$$

Intuitively, the Information Gain thus measures how much the impurity or uncertainty is reduced making a distinct split. Thereby, $H(T)$ is the impurity before the split and the sum is the impurity after the split. One problem of Information Gain is that there is a bias towards multivalued attributes.

One of the advantages of decision trees is that they do not need much data preparation. Decision trees can handle nominal or ordinal and numerical data out of the box. For handling numerical data the construction algorithm just tries every possible split for the present range of feature values.

A disadvantage of decision trees is that they tend to overfit if the tree is not pruned at all.

2.6.5 Random Forests

The random forest algorithm [Breiman, 2001] constructs several decision trees from the training data. In fact, the algorithm splits the training set randomly into a number of smaller training sets and constructs one decision tree for each split. This technique is called bagging. Also, during construction of the trees at each node generation step only a random subset of the features is being regarded. This technique is called random split selection. To predict a label, the output of all of the created trees is combined. Since the random forest algorithm bases its decision on several decision trees, and thus several distinct classifiers, the algorithm falls into the category of ensemble classifiers. Both bagging and random split selection reduce the risk of overfitting making random forest a good and often superior alternative to using just one decision tree.

2.6.6 Support Vector Machines

Support Vector Machines (SVMs) try to separate the data at hand by a hyperplane. A hyperplane's dimension is one lower than the dimension of the data. In case of two-dimensional data the hyperplane is thus a line (see Figure 2.7 for a visualization). The hyperplane is chosen that has the largest margin to the two classes to separate. Note that a SVM can only separate two classes natively. Multi-class classification is done in a one-vs-all strategy, hence fitting one classifier per class. Since the margin to the classes is maximized the classifier is a maximum-margin classifier. The name Support Vector Machine stems from the support vectors that define the chosen hyperplane. If the data is not linearly separable the data can still be separated using the kernel trick, which means projecting the data into a higher dimensional space.

2.6.7 Evaluation

To evaluate a classifier it is advisable to use a test set with instances whose class is known. Then, the predictions of the classifier on the test set can be compared to the true labels. In the following

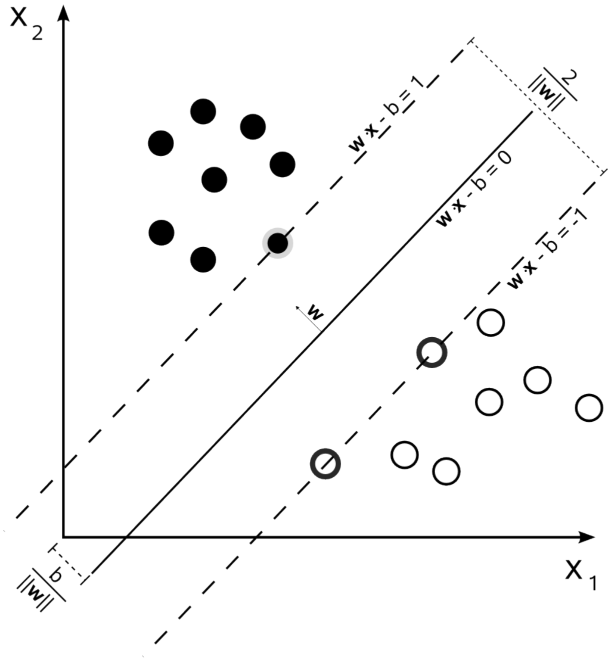


Figure 2.7: Example of a maximum-margin hyperplane/line splitting two classes

we assume that there is a class c_1. If the classifier predicts class c_1 for an instance and the instance is in fact labeled as c_1, the output is called a true positive (TP). Not predicting c_1 when in fact the true label is c_1 yields a false negative (FN). If an instance is classified as belonging to c_1, but it does not, this is called a false positive (FP). A true negative (TN) means that the instance does not belong to c_1 and that is what the classifiers predicts, too. Table 2.6 summarizes the cases described.

		Predicted class:	
		c_1	$\neg c_1$
True class:	c_1	TP	FN
	$\neg c_1$	FP	TN

Table 2.6: Confusion matrix

Based on the number of true positives, false negatives, false positives and true negatives, a number of measures can be defined, which all provide a rating between 0 and 1. A straight-forward metric is the accuracy, which is the percentage of instances whose prediction is correct. Accuracy is not always a good evaluation metric. For instance, sometimes it is very important to either have few FPs or to have few FNs. Then recall and precision come in handy. Recall is the fraction of the objects that belong to c_1 that are classified as such. Precision is the fraction of the objects classified as c_1 that truly belong to c_1. It is easy to obtain a recall of 1: Just classify all instances as c_1. It is also easy to obtain a precision of 1: Just classify no instances as c_1. Based on these observations it makes sense to define another measure: The F-score. It combines recall and precision. In case of $\beta = 1$ it is the harmonic mean of recall and precision. Note that in the following we write F-score if we mean F_β-score. Table 2.7 provides an overview of the described measures.

Metric	**Definition**
Recall	$\frac{TP}{TP+FN}$
Precision	$\frac{TP}{TP+FP}$
Accuracy	$\frac{TP+TN}{TP+FP+TN+FN}$
F_β-Measure	$\frac{(1+\beta^2)\cdot Recall\cdot Precision}{(\beta^2\cdot Precision)+Recall}$

Table 2.7: Metrics for evaluating a classifier's performance

In our example from above we deal with a binary classifier. A binary classifier decides if an object belongs to a class (in our example class c_1) or not. If there are multiple classes, recall, precision and F-score can be calculated for every class. There are different ways to get one final F-score for the multi-class case. The F-score can be calculated globally, which means counting the total number of TPs, FNs and FPs and directly computing the F-score from these numbers. This yields a so called micro-averaged F-score. Another way to receive one F-score in the presence of multiple classes is combining the F-scores of the single classes. The macro-averaged F-score is the (weighted or unweighted) mean of the F-scores for the single classes. If we provide a F-score in the following, we either refer to the binary case or are in fact providing a macro-averaged weighted F-score.

It is often difficult to obtain labeled data. The acquired data then again has to be split into training and test data. The smaller the training set and the test set are, the less reliable are the results of the evaluation. To get reliable evaluation results a cross-validation is often used. For a cross-validation the labeled data is split N times. Now, evaluation scores are computed N times for N different training and test sets. The test set is one of the N folds. The training data is made up of the rest $N-1$ folds. In the end, evaluation scores can be reported taking the mean of the scores for the N runs.

3 Previous Work

Methods for the identification of antonymy can roughly be categorized into three classes. Some of the methods deal with distinguishing antonymy from synonymy, other methods try to measure the degree of contrast between two words. Another branch of research is the mining of antonym pairs from text. Each of the three classes gets their own section in this chapter. Lots of the approaches for the identification of antonymy use patterns to reach the goal. Thus, we conclude this chapter with a section on general pattern-based approaches for the discovery of semantic relations between words.

3.1 Distinguishing Antonyms from Synonyms

Often the research goal is to distinguish antonyms from synonyms, a more specific task than descriminating antonyms from non-antonyms. The eligability of this task is motivated by the common claim, that measures of distributional similarity normally cannot distinguish synonyms from antonyms [Mohammad et al., 2013].

Some researchers use pattern-based approaches to differentiate between antonymy and synonomy, using either hand-coded [Lin et al., 2003], or automatically extracted patterns [Schulte im Walde and Köper, 2013, Turney, 2008]. Others introduce new distributional measures to distinguish synonyms from antonyms in an unsupervised manner [Santus et al., 2014, Scheible et al., 2013].

It is difficult to compare classification results, since lots of different datasets are used.

Lin et al. [2003] present two methods to retrieve synonyms from distributionally similar words. The first method makes use of two patterns [*either* <X> *or* <Y>] and [*from* <X> *to* <Y>]. It relies on the hypothesis that if two words X and Y appear in one of the patterns, they are very unlikely synonymous. At this point, we will not discuss the second method, which relies on bilingual dictionaries, since it suffers from a very low recall.

Lin et al. [2003] evaluate their methods on a custom dataset containing 80 randomly drawn pairs of synonyms and 80 randonmly drawn pairs of antonyms from Webster's Collegiate Thesaurus [Kay, 1988].

The pattern-based approach yields a precision of 86.4% and a recall of 95%. These numbers look promising at first. But the antonyms in the dataset of Lin et al. [2003] were selected from a list of high-frequency terms. Mohammad et al. [2013] claim, that Lin Patterns have a low coverage for their antonym set.

Turney [2008] proposes a unified approach to analogies, synonyms, antonyms, and associations.

The approach consists of the PairClass algorithm (as refered to by the author), a feature extraction algorithm, and supervised classification based on a Support Vector Machine (SVM). For each word pair, PairClass extracts a feature vector, which consists of the logarithmic frequencies of the top k *

N patterns containing the word pair. k was set to 20; N denotes the number of word pairs. Before applying supervised classification, the feature vectors are normalized to unit length.

Turney [2008]'s evaluation is based on a set of 136 ESL questions. He reports an accuracy of 75% as a result of a ten-fold cross-validation. However, his dataset is not stratified. According to him, always guessing the majority class would lead to an accuracy of 65.4%.

Schulte im Walde and Köper [2013] present a pattern-based approach for distinguishing semantic relations for German words. They created a custom dataset drawn from GermaNet [Hamp and Feldweg, 1997], containing word pairs consiting of antonymous, synonymous and hyponymous nouns, verbs and adjectives. Each of the classes (e.g. antonymous nouns or synonymous adjectives) contains at least 61 and at most 97 pairs.

The authors extracted lexico-syntactic patterns between the word pairs and computed pattern-frequency vectors. Using a nearest-centroid classifier they report an F-score of 70.75% on the task of distinguishing antonyms from synonyms. Interestingly, non of the generalisations of the patterns, e.g. replacing determiners with their POS-tag, which were tried by the authors, improved the result.

The research goal of Scheible et al. [2013] is to show, that adjectival synonyms and antonyms can be distinguished via a word space model by introducing a new distributional measure. Instead of taking into account all words in a window of a certain size for feature extraction, they experiment with only taking into account words of a certain part-of-speech.

The intuition behind that is demonstrated on the example of the "happy/sad man" or the "unhappy/sad man". Whilst the synonym pair unhappy/sad and the antonym pair happy/sad are both likely to co-occur with the noun man, the set of verbs co-occuring with "happy/sad man" will most likely be different from the set of verbs co-occuring with "unhappy/sad man". Thus, Scheible et al. [2013] basically suggest to characterize an adjective by the verbs, which most likely co-occur with it.

Scheible et al. [2013]'s approach achieves 70.6% accuracy in the antonym/synonym classification task on a dataset consiting of 97 antonym pairs and 97 synonym pairs from GermaNet.

Another distributional measure, *APAnt*, to distinguish antonymy from synonymy is presented in [Santus et al., 2014]. *APAnt* is based on the observation that antonyms are often very similiar except in one dimension of meaning. As one example Santus et al. [2014] present dwarf and giant. Both are most likely persons, e.g. having two legs, two feet and so on, thus beeing very similar except in their size. The authors furthermore claim, that the contrasting dimension of meaning is a very salient one. For example dwarf will often co-occur with words like small or little and giant will often co-occur with words like big or huge.

APAnt is an adaption of the Average Precision measure. It compares the N most salient contexts of a pair of antonyms and synonyms. Santus et al. [2014] hypothesise, that synonyms share a number of salient contexts that is significantly higher than the ones shared by antonyms.

APAnt is evaluated on a dataset containing more than 1000 antonymy-related and 1000 synonymy related word-pairs obtaining an accuracy of 73% percent on the task of distinguishing antonyms from synonyms.

3.2 Computing Word-Pair Antonymy

Methods to compute word-pair antonymy range from the direct utilization [Mohammad et al., 2008] of WordNet [Miller, 1995] to extending Latent Semantic Analyis [Yih et al., 2012] and training word embeddings specific to the task of antonymy detection [Ono et al., 2015].

Antonymy identification is mostly evaluated on the GRE antonym question task, with F-scores ranging from 60% up to 90%. The GRE dataset consists of over 1000 closest-opposite questions. A closest-opposite question consists of a target word, which has to be mapped to the word out of five given words, which is the closest opposite of the target.

Mohammad et al. [2008] assume, that all word-pairs of contrasting thesaurus categories are indeed antonyms. Contrasting thesaurus categories are computed by determining if one category contains a word, which can be formed by concatenating a word from the other thesaurus category with affixes like -in or -un, which commonly indicate antonymy.

On the GRE dataset, Mohammad et al. [2008]'s method achieves an F-score of at most 70%.

Yih et al. [2012] introduce a word vector space, which incorporates synonym and antonym information from thesauri, where antonyms lie on opposite sides of a sphere.

Yih et al. [2012] report an F-score of 87% on the GRE dataset.

Ono et al. [2015] train word embeddings specific to the task of identifying antonymy. They use both WordNet and Roget to obtain synonym and antonym pairs for the supervised training of their word embeddings. Their objective function, which is maximized to train the word embeddings, is designed to produce similar embeddings for synonyms and embeddings with low similarity for antonymous word pairs.

Choosing the response with the lowest similarity score of the respective word embeddings to the target word on the GRE dataset, Ono et al. [2015] report an F-score of 89%.

3.3 Antonym Extraction

Apart from classifying a word pair as either antonymous or not or measuring the contrast of two given words by comparing word embeddings another approach is to generate a list of antonymous word pairs. Of course the resulting list can also be used to solve classification tasks as Mohammad et al. [2008] do with their list generated from lexical resources on the GRE task. Lobanova [2012] extract antonyms from Dutch text using first plain text patterns, second part-of-speech patterns and third dependency patterns. The patterns are extracted using a seed set of antonymous word pairs and then ranked by reliability meaning precision of extraction. Their evaluation hints that there is no improvement with dependency patterns over surface part-of-speech patterns and they recommend part-of-speech patterns for antonym extraction.

Several of the methods presented previously for the automatic identification of antonymy rely on pattern information. Lin et al. [2003] used two hand-coded patterns to discover antonyms and distributionally similar pairs of words, Turney [2008] described a pattern-based approach to differentiate between different semantic relations including antonymy, Schulte im Walde and Köper [2013] distinguished between the relations hyponymy, synonymy, and antonymy of German word pairs using patterns and Lobanova [2012] compared different pattern-based approaches for Dutch antonym extraction. Since patterns play an important role in research on antonymy identification, we will shortly cover other research on patterns for the identification of semantic relations other than antonymy.

[Hearst, 1992] was the first who used patterns for the automatic mining of hyponym relations. Like Lin et al. [2003], who later presented two patterns for the identification of antonyms, Hearst [1992] introduced a handful of patterns to extract hypernym/hyponym pairs from text. See Table 3.1 for the patterns used.

Pattern	**Example**
NP_0 such as $\{NP_1, NP_2$ ..., (and | or)$\}$ NP_n	"The bow lute, such as the Bambara ndang, ..."
such NP as $\{NP$, $\}$* $\{$(or | and)$\}$ NP	"... works by such authors as Herrick, Goldsmith, and Shakespeare."
NP $\{$, $NP\}$* $\{$,$\}$ or other NP	"Bruises, wounds, broken bones or other injuries ..."
NP $\{$, $NP\}$* $\{$,$\}$ and other NP	"... temples, treasuries, and other impor- tant civic buildings."
NP $\{$,$\}$ including $\{NP$,$\}$* $\{$or | and$\}$ NP	"All common-law countries, including Canada and England ..."
NP $\{$,$\}$ especially $\{NP$,$\}$* $\{$or | and$\}$ NP	"... most European countries, especially France, England, and Spain."

Table 3.1: Hearst patterns and examples (adapted from [Hearst, 1992])

Snow et al. [2005] generalized Hearst's approach. They collected noun pairs from corpora and then labeled them „hypernym pair“ or „no hypernym pair“. Then, they extracted all the dependency paths possible between the noun pairs. On the pairs, their labels and information about with which of the dependency paths the pairs were connected they trained a hypernym classifier. The classifier learned which of the possible dependency paths were indicators for hypernymy. Among the paths with the best precision-recall trade-off for finding hypernym pairs were the paths described by the Hearst patterns.

4 Methods

In this chapter, our approach on learning whether words are antonyms or not is described. We follow a supervised methodology whose general setting is described in Section 4.1. For supervised learning training data is required and features have to be defined. Thus, this chapter contains both a section on acquiring training data and a section on feature engineering. The chapter is concluded with a section on finding the best feature combination and deciding on which classifier to use.

4.1 General Setting

We trained separate classifiers for the part-of-speech categories adjectives, nouns and verbs. The classifiers are trained on instances of the training data with antonym pairs and non-antonym pairs from WordNet and extracted features from several sources. We cross-validated our classifiers on the data acquired from the lexical resource as one mean to not overfit the training data and later evaluate our model trained on the whole training data on different evaluation sets from literature. A basic overview of the process gives Figure 4.1.

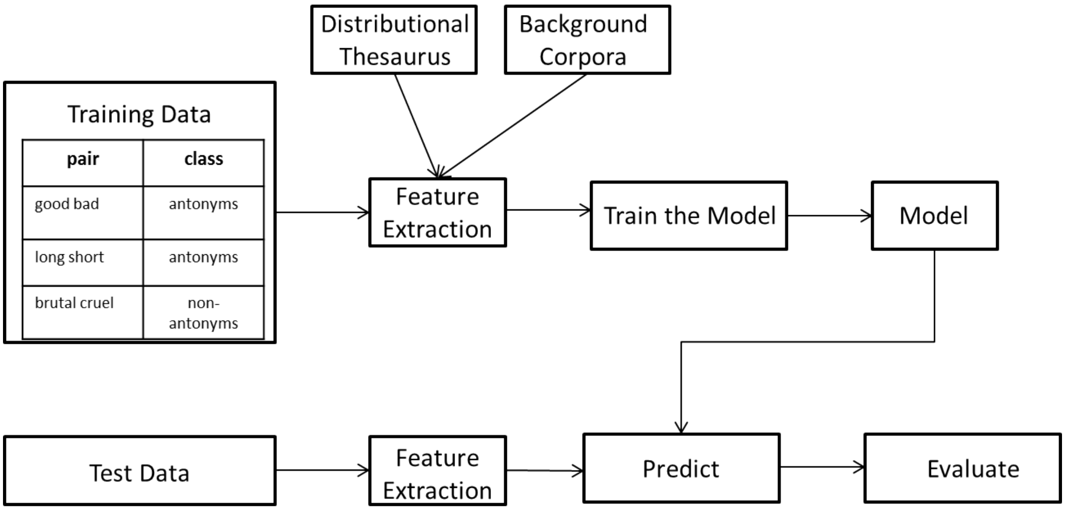


Figure 4.1: Supervised Classification

4.2 Training Data

To our best knowledge no English data set exists that fulfills the following criteria: The data set contains at least antonymous and synonymous word pairs and pairs related by hyponymy. The data set is large enough for training meaning it contains at least 100 word pairs per class. Hence, we compiled a new data set from WordNet.

When compiling the data sets we took care of the following: We did not include word pairs containing a word that occurs less than 100 times in a 105 million sentences corpus consisting of news articles in the respective language. This threshold should make sure, that we can learn something useful from a word pair, and that what is learned is not based upon unusual random occurrences of the pair's words. We made sure that we did not include one word more than once in the training data. This helps to prevent the classifiers from learning that a word is e.g. a prototypical antonym or non-antonym. For example if we had several pairs containing the word “bad” in our training set and all pairs had the class label antonym, the following might happen. We split the data for a cross-validation such that we have a pair containing ”bad“ in the training set and the test set. Now, the classifier might just learn that “bad” is a prototypical antonym and we could get good test results in spite our classifier did not learn what generally indicates antonymy. But, since we only include a word in at most one pair of the whole data, this cannot happen. Every possible split of our data is a delexicalized one. See [Levy et al., 2015] for more information on delexicalization of the training data for learning semantic relations.

Which relations were included in the data set was based on the information available in WordNet. WordNet provides information about nouns, adjectives and verbs and about relations such as synonymy, antonymy, hypernymy/hyponymy and holonymy/meronymy. Co-hyponyms can be extracted with the help of hypernymy/hyponymy relations. WordNet does not contain hypernymy/hyponymy and holonymy/meronymy information for adjectives and verbs. Thus our adjective and verb data contains only synonymous, antonymous and unrelated word pairs. We determined unrelated words by extracting words with a high path length between the word's synsets.

We stratified our data sets to get equal prior probabilities for each class, antonyms and non-antonyms. Finally, we obtained a data set we will refer to as *WordNetPairs* (see also Table 4.2).

	ANT	SYN	HYP	COHYP	MER	UNR
			WordNetPairs			
NOUN	466	94	93	93	93	93
ADJ	999	450				449
VERB	310	155				155

Table 4.2: Training data

4.3 Feature Engineering

For corpus-based feature extraction we used different corpora whereby each of them fulfilled a distinct and special need of the respective feature extraction algorithm. For one, we used a corpus of random sentences from English news texts from the years 2005 to 2010, *News105M*. Each of the sentences was POS-tagged and lemmatized and also parsed with a dependency parser. In case full texts (meaning coherent sentences) were needed, we used a corpus consisting of news articles, the *AQUAINT-2* corpus [Vorhees and Graff, 2008]. Furthermore, we worked with the Google N-gram corpus [Brants and Franz, 2006]. This corpus delivers unigram to 5-gram counts compiled from over 95 billion sentences, thus being by far the largest corpus used. Apart from these corpora, we used information about words that was gained in an unsupervised training process. First, we used a DT, computed with the JoBimText-Framework on *News105M*. Second we used word embeddings trained by Mikolov et al. [2013]. Not a single feature extraction algorithm is knowledge-based meaning it uses information from manually tagged resources (as e.g. WordNet).

4.3.1 Morphology

Morphological clues, prefixes and suffixes indicating antonymy, are easy to spot and precise (for instance on the adjective data we attained a precision of about 90% of the morphological clues). Unfortunately, only a subset of all antonym pairs comes with these indicators. We used the morphological pattern rules from Mohammad et al. [2013], which are visualized in Table 4.3. Then, we incorporated one binary feature indicating whether one of the morphological antonym patterns matches a word pair or not.

PatternID	**Word 1**	**Word 2**	**Example Pair**
1	X	antiX	clockwise-anticlockwise
2	X	disX	interest-disinterest
3	X	imX	possible-impossible
4	X	inX	consistent–inconsistent
5	X	malX	adroit–maladroit
6	X	misX	fortune–misfortune
7	X	nonX	aligned–nonaligned
8	X	unX	biased–unbiased
9	lX	illX	legal–illegal
10	rX	irrX	regular–irregular
11	imX	exX	implicit–explicit
12	inX	exX	introvert–extrovert
13	upX	downX	uphill–downhill
14	overX	underX	overdone–underdone
15	Xless	Xful	harmless–harmful

Table 4.3: Morphological antonym pattern rules

Apart from the hand-coded patterns, we added three more features to address morphological clues. For one, we added another binary feature specifying if one of the words is a part of the other one. Also, we split the two words into their syllables[1] and added the Jaccard distance (see Equation 4.1) of the two syllable sets as feature.

[1] We used PyHyphen for this task: `https://pypi.python.org/pypi/PyHyphen/`

$$Jaccard(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{4.1}$$

Furthermore, we added one feature carrying the normalized Levenshtein distance [Levenshtein, 1966] of the two words. The Levenshtein distance denotes how many deletions, insertions or substitutions are needed to transform one word into the other. To get a normalized count between 0 and 1 the distance is divided by the word length of the longest word of the pair.

The features of this subsection are later referred to as feature category *morpho*.

4.3.2 Distributional Similarity

If a word pair is antonymous it is very likely that one word is present in the top distributionally similar words of the other (at least regarding distributional similarity computed with the JoBimText framework using standard parameters). For instance "bad" is the most distributionally similar word of "good" (see Table 4.4 for the top 10 most distributionally similar words of "good").

Word	Similarity score
bad	112
excellent	99
decent	79
great	72
solid	65
strong	56
poor	55
tough	55
outstanding	54
terrific	54

Table 4.4: The top distributionally similar words for "good"

This does not always hold and the most distributionally similar word of a word often is in fact a synonym. But our experiments show that incorporating features that state whether a word is present in the top n, with $n \in \{1,3,5,10,20\}$, distributionally similar words helps our classifiers. Figure 4.2 shows how often it is the case for antonymous and synonymous adjectival word pairs that one word occurs in the top n similar words of the other.

Mohammad et al. [2013] also researched how antonymy and synonymy affects distributional similarity. They used the definition of distributional similarity in Lin [1998] to compute similarity scores on data drawn from WordNet. In their studies, antonym pairs had an average similarity of 0.064, random pairs of 0.036 and synonym pairs of 0.056. These results also underpin the hypothesis that antonyms are even more distributionally similar than synonyms.

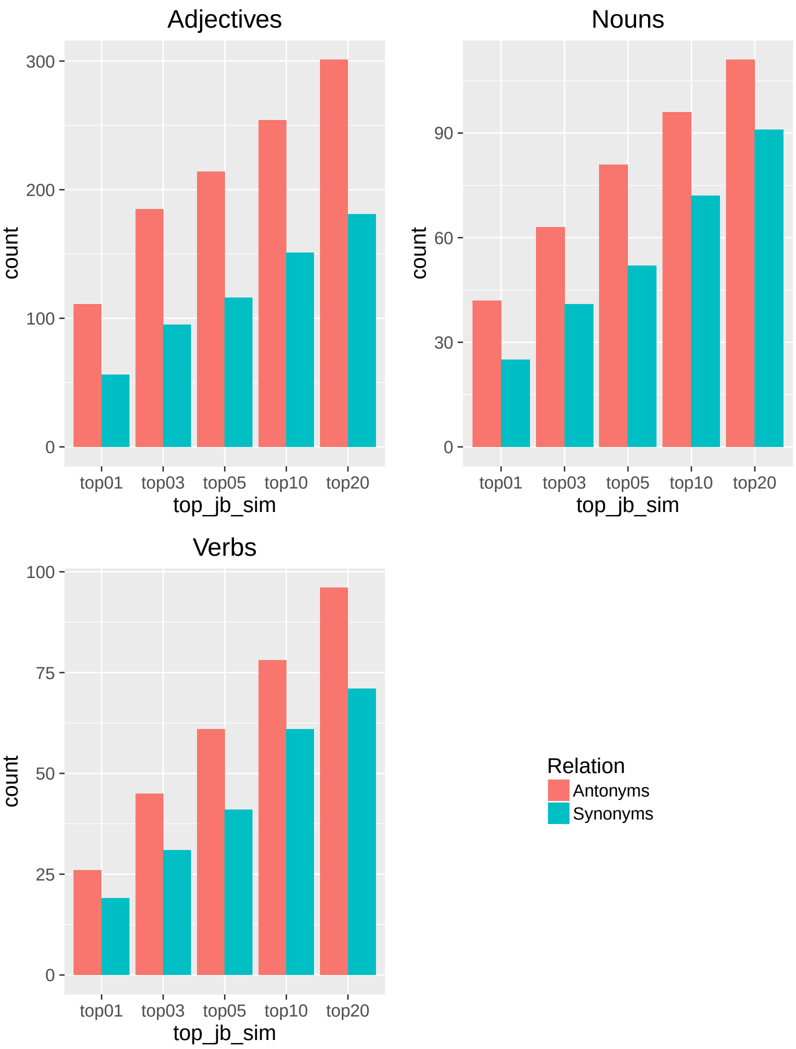


Figure 4.2: Top distributional similarity for antonym and synonym pairs

Intuitively, this makes sense, since antonymy mostly denotes very high similarity of two things in all except one dimension. Santus et al. [2014] illustrate this idea on the example of a dwarf and a giant. Both are persons sharing plenty of properties, except one is small and the other big.

The distributional features computed with the help of the JoBimText framework are later referred to as feature category *jb-sim*. We also included the cosine similarity of the SKIP-gram word embeddings [Mikolov et al., 2013] of the two words as another measure of distributional similarity. Word embeddings represent words as vectors, that are computed with neural networks on word transition probabilities. The word embedding similarity of the words is later referred to as feature category *word-embed*.

4.3.3 Co-occurence Probabilities

Charles and Miller [1989] proposed that antonyms relatively frequently co-occur in the same sentence. This hypothesis was later backed by evidence in several experiments. Justeson and Katz [1991] tested the hypothesis with a list of 35 antonyms stating that in a corpus of 25 million words, all pairs yield highly significant numbers of co-occurrences. Fellbaum [1995] also found supporting evidence testing with 47 antonym pairs on the Brown corpus [Francis and Kucera, 1979]. Mohammad et al. [2013] computed PMI scores of antonyms, synonyms and random pairs from WordNet. The average PMI scores of antonyms were significantly higher than the ones of synonyms and random pairs (see also Table 4.5).

Type of relation	Average PMI
antonyms	1.471
synonyms	0.412
random pairs	0.032

Table 4.5: PMI scores of antonyms, synonyms and random pairs from WordNet (adapted from Mohammad et al. [2013])

We computed LMI scores for all antonymous and synonymous pairs in *WordnetPairs* regarding a window of one sentence using *News105m* as a corpus.

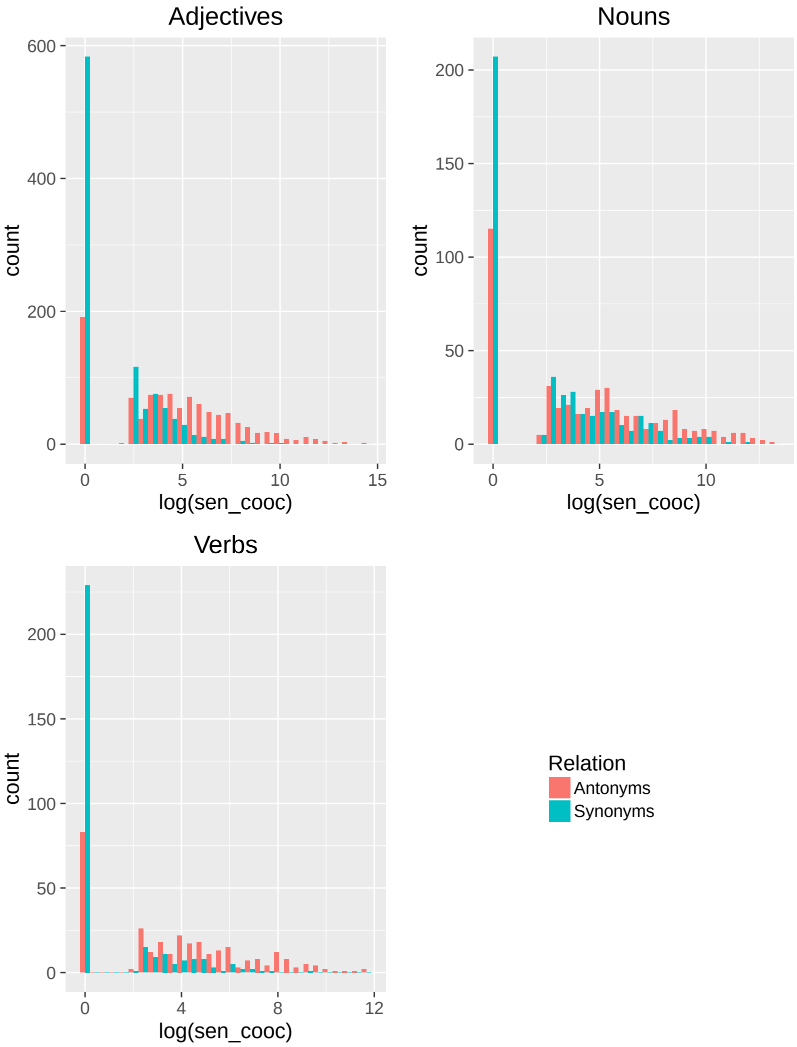


Figure 4.3: Intra-sentence association scores of antonyms and synonyms

We chose LMI over PMI to not involuntarily prefer infrequent pairs. The resulting scores can be seen in Figure 4.3. Note that we mapped non-existing LMI scores, due to pairs never co-occurring in a sentence, to zero and that we plotted the logarithm of the actual LMI scores incremented by one to fit the x axis into a range from zero to fifteen. The figure clearly shows that antonym pairs have on average higher association scores than synonyms. 583 of 999 synonym pairs in our data set never occur together in one sentence in *News105m*. In contrast only 191 of 999 antonym pairs do not co-occur at least once in a sentence.

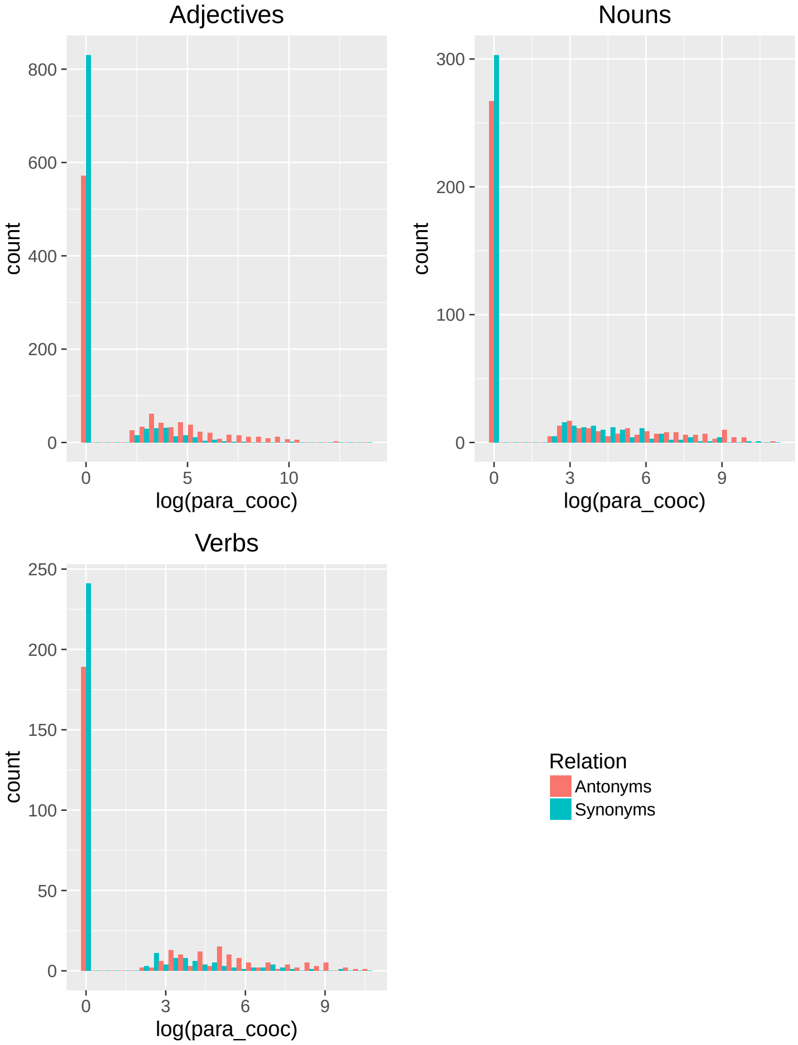


Figure 4.4: Intra-paragraph association scores of antonyms and synonyms

However, words can occur together in a sentence significantly more often than chance because of several reasons. Word pairs from multi-word expressions or collocations or word pairs that are in any other form semantically related can have high association scores, too. Thus, LMI scores of a word pair can only serve as one feature separating antonyms well from unrelated words and not as well from synonyms. A cross-validation on our antonymy/synonymy data from WordNet using only this feature and Decision Trees leads to an average F-score of about 0.68.

We later refer to the intra-sentence association scores as feature category *sen-cooc*.

Having the intuition in mind that synonyms could occur more often than antonyms in different sentences of a paragraph since writers do not want to repeat words, we also computed LMI scores for paragraph windows. Therefore, we counted the number of times the word pairs co-occurred in a window of three sentences excluding them co-occurring in the same sentence in the *AQUAINT-2* corpus. The resulting scores are visualized in Figure 4.4.

Unfortunately, our paragraph co-occurence hypothesis does not hold. Antonyms still occur more often together in a paragraph then synonyms. But the paragraph co-occurrence scores are apparently still slightly useful as a feature for our classifiers since cross-validating on the WordNet antonym/synonym data yields a F-score of 0.57.

We later refer to the intra-paragraph association scores as feature category *para-cooc*.

4.3.4 Association Scores of Pairs with Context Words

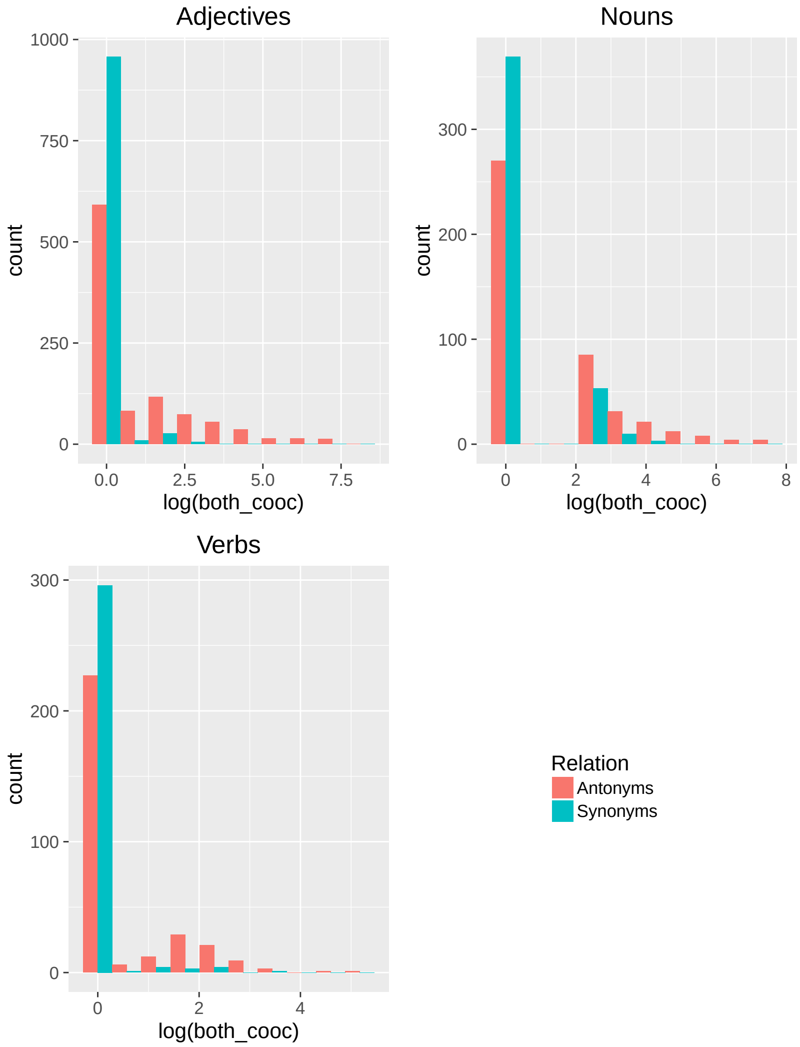


Figure 4.5: LMI scores of antonymous and synonymous pairs with “both”

Following work on lexical substitution from Szarvas et al. [2013], we added LMI scores of the pairs with words occurring together with the pairs in a sentence as features. This is an easy way to incorporate syntactic information into our model. A word pair that often occurs in the pattern [*both* <X> *and* <Y>] will produce high association scores with the words "both" and "and". Also, a pair that often occurs in the pattern [<X> *or* <Y>] will produce high association scores with "or". Figure 4.5 shows that antonyms tend to be more associated with „both". At the beginning we included association scores of the pairs with 10000 very frequent words in *News105m*, later we settled only for words that occured in patterns highly associated with the antonyms in our training set.

The features of this subsection are later referred to as feature category *news105-context*.

The next step is not to measure association of pairs with words in their context but to measure the association of pairs with patterns connecting them in a sentence.

4.3.5 Lin Patterns

As mentioned in section 3.1 Lin et al. [2003] proposed two patterns to distinguish synonyms from antonyms, [*either* <X> *or* <Y>] and [*from* <X> *to* <Y>]. Simplifying their pattern-based method they marked a pair as synonymous if it did not occur or only seldomly occurred with the two patterns using search engine query results (for a more complete explanation of the method we refer to section 3.1). Lin et al. [2003] reported a precision of 86.4% and a recall of 95% on a data set containing 80 pair of synonyms and 80 pairs of antonyms. The reported scores combine to a F-score of about 90%. As the experiments were conducted with a search engine (even a search engine that is not existent any more, by name: AltaVista), they can not be reproduced precisely. We decided to check the usefulness of the Lin Patterns not with a recent search engine but with regex searches in two corpora: *News105M* and the Google N-gram corpus [Brants and Franz, 2006]. 161 of the antonym pairs from *WordNetPairs* do occur in the pattern [*either* <X> *or* <Y>] in *News105M*, only one synonym pair co-occurs with the pattern. 358 antonym pairs co-occur with [*either* <X> *or* <Y>] in the Google N-gram corpus and only two synonym pairs. 440 of the antonym pairs appear in at least one of the Lin patterns in at least one of the corpora. This holds for only 9 synonym pairs. Table 4.6 and Table 4.7 provide counts for the antonym and synonym pairs regarding all possible combination of patterns and corpora.

		Corpus		
		News105m	Google N-gram	both
Pattern	[*from* <X> *to* <Y>]	180	115	212
	[*either* <X> *or* <Y>]	161	358	390
	both	245	385	440

Table 4.6: Number of times antonym pairs occurred with Lin patterns

From our findings we conclude that Lin patterns are very precise but do not offer a high recall. 559 of the 999 antonym pairs can not be found in at least one pattern in the combined corpora.

We later refer to the association scores of pairs with Lin patterns in *News105M* as *news105-lin*.

		Corpus		
		News105m	Google N-gram	both
Pattern	[*from* <X> *to* <Y>]	1	5	6
	[*either* <X> *or* <Y>]	1	2	3
	both	2	7	9

Table 4.7: Number of times synonym pairs occured with Lin patterns

4.3.6 Part-of-Speech Patterns

We relied on the work of Lobanova [2012] and choose to incorporate part-of-speech pattern information into our classification process. Lobanova [2012] compared surface textual patterns, surface part-of-speech patterns and dependency patterns for automatic opposite extraction. She claims to not have found overall improvement with dependency patterns over surface part-of-speech patterns and recommends part-of-speech patterns for opposite extraction. Unlike Lobanova [2012], we do not automatically extract a list of opposites. Instead, we train classifiers following the methodology of Snow et al. [2005] who trained a hyponymy-only classifier. We opt to do the same for antonymy. The intuition behind Snow et al. [2005]'s and our approach is that if word pairs in a small list of a designated relation are highly associated with some patterns, other word pairs that are also highly associated with these pattern are likely to be related in the same way.

At first, we extracted all patterns that co-occurred at least once with antonyms from our training data sets in *News105m*. Table 4.8 lists the patterns that most often occur with adjective antonyms from our training set.

Pattern	**Occurences**
<ANT>/JJ and/CC <ANT>/JJ	92456
<ANT>/JJ or/CC <ANT>/JJ	21018
the/DT <ANT>/JJ and/CC <ANT>/JJ	12062
<ANT>/JJ and/CC <ANT>/JJ ,/,	11000
,/, <ANT>/JJ and/CC <ANT>/JJ	8220
<ANT>/JJ and/CC <ANT>/JJ ./.	8082
of/IN <ANT>/JJ and/CC <ANT>/JJ	7895
both/DT <ANT>/JJ and/CC <ANT>/JJ	7609
between/IN <ANT>/JJ and/CC <ANT>/JJ	6336
<ANT>/JJ ,/, <ANT>/JJ	6212
<ANT>/JJ or/CC <ANT>/JJ ,/,	4607
in/IN <ANT>/JJ and/CC <ANT>/JJ	4054
,/, <ANT>/JJ and/CC <ANT>/JJ ,/,	3650
<ANT>/JJ or/CC <ANT>/JJ ./.	3646
,/, <ANT>/JJ or/CC <ANT>/JJ	3118
<ANT>/JJ ,/, <ANT>/JJ and/CC	2924
on/IN <ANT>/JJ and/CC <ANT>/JJ	2529

Pattern	Occurences
,/, both/DT <ANT>/JJ and/CC <ANT>/JJ	2003
<ANT>/JJ ,/, <ANT>/JJ ,/,	1895
for/IN <ANT>/JJ and/CC <ANT>/JJ	1811
,/, <ANT>/JJ or/CC <ANT>/JJ ,/,	1803
a/DT <ANT>/JJ and/CC <ANT>/JJ	1759
<ANT>/JJ to/TO <ANT>/JJ	1705
both/DT <ANT>/JJ and/CC <ANT>/JJ ,/,	1496
runners/NNS on/IN <ANT>/JJ and/CC <ANT>/JJ	1468
in/IN the/DT <ANT>/JJ and/CC <ANT>/JJ	1463
<ANT>/JJ and/CC the/DT <ANT>/JJ	1438
between/IN the/DT <ANT>/JJ and/CC <ANT>/JJ	1409
the/DT <ANT>/JJ or/CC <ANT>/JJ	1388
to/TO <ANT>/JJ and/CC <ANT>/JJ	1371
with/IN <ANT>/JJ and/CC <ANT>/JJ	1358
<ANT>/JJ and/CC <ANT>/JJ and/CC	1354
's/POS <ANT>/JJ and/CC <ANT>/JJ	1346
a/DT <ANT>/JJ or/CC <ANT>/JJ	1315
the/DT <ANT>/JJ and/CC the/DT <ANT>/JJ	1315
<ANT>/JJ and/CC <ANT>/JJ in/IN	1295
,/, <ANT>/JJ and/CC <ANT>/JJ ./.	1269
<ANT>/JJ time/NN <ANT>/JJ	1266
the/DT <ANT>/JJ time/NN <ANT>/JJ	1262
for/IN the/DT <ANT>/JJ time/NN <ANT>/JJ	1182

Table 4.8: Patterns most commonly found with adjectival antonym pairs from training set

Short patterns like [<ANT>/JJ *or/CC* <ANT>/JJ] and [<ANT>/JJ *and/CC* <ANT>/JJ] occur most often with the adjectival antonyms from the WordNet data.

[*both/DT* <ANT>/JJ *and/CC* <ANT>/JJ] and [*between/DT* <ANT>/JJ *and/CC* <ANT>/JJ], which we later show are also precise patterns for antonymy identification also among the most occurring patterns.

Figure 4.6 shows how often the patterns occur in total in the corpus. This distribution of patterns follows the typical power-law distribution of language elements. Lots of the patterns occur very infrequently and only some occur very frequently. This typical distribution of language elements was discovered by Zipf [1929] and also applies for example to the frequency of words.

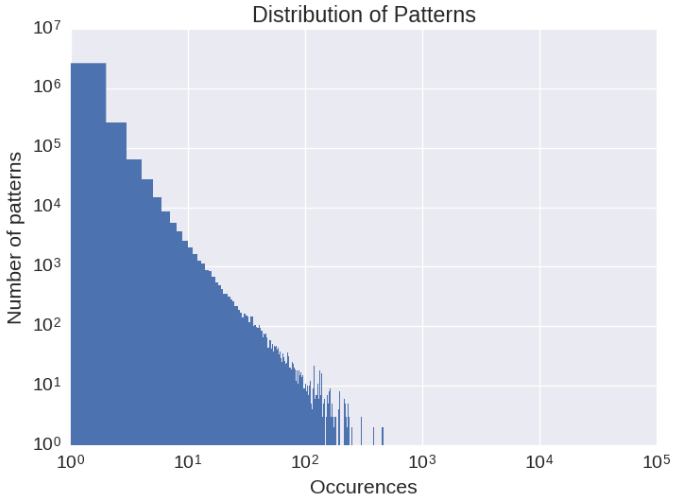


Figure 4.6: Distribution of patterns

The extracted patterns were ranked by their association scores with the antonym pairs from the training set. The top ranked antonym patterns for adjectives are listed in Table 4.9, the ones for nouns in Table 4.10 and the ones for verbs in Table 4.11. [<ANT>/XX *and/CC* <ANT>/XX] is among the top two patterns for XX $\in JJ, NN, VB$. Furthermore, [<ANT>/XX *or/CC* <ANT>/XX] is among the top two patterns for XX $\in JJ, NN, VB$. Other interesting patterns are [*both/DT* <ANT>/XX *and/CC* <ANT>/XX] and [*between/DT* <ANT>/XX *and/CC* <ANT>/XX]. Lin patterns do not turn up in any of the top ten ranked patterns.

Pattern	**LMI score**
<ANT>/JJ or/CC <ANT>/JJ	1338.9366
<ANT>/JJ and/CC <ANT>/JJ ./.	1100.8132
<ANT>/JJ or/CC <ANT>/JJ ./.	820.1146
between/IN <ANT>/JJ and/CC <ANT>/JJ	766.0270
<ANT>/JJ and/CC <ANT>/JJ ,/,	739.7385
of/IN <ANT>/JJ and/CC <ANT>/JJ	612.6060
the/DT <ANT>/JJ and/CC <ANT>/JJ	561.0529
both/DT <ANT>/JJ and/CC <ANT>/JJ	507.9798
<ANT>/JJ or/CC <ANT>/JJ ,/,	411.0576
,/, <ANT>/JJ and/CC <ANT>/JJ	353.8351
between/IN <ANT>/JJ and/CC <ANT>/JJ ./.	283.9915
,/, <ANT>/JJ and/CC <ANT>/JJ ./.	275.2062
,/, <ANT>/JJ or/CC <ANT>/JJ ./.	271.1614
in/IN <ANT>/JJ and/CC <ANT>/JJ	245.4885
on/IN <ANT>/JJ and/CC <ANT>/JJ	239.0602
,/, <ANT>/JJ or/CC <ANT>/JJ	238.8114
,/, <ANT>/JJ and/CC <ANT>/JJ ,/,	227.0190
both/DT <ANT>/JJ and/CC <ANT>/JJ ./.	217.2118

<ANT>/JJ 1980s/NNS and/CC <ANT>/JJ	216.4160
the/DT <ANT>/JJ 1980s/NNS and/CC <ANT>/JJ	216.4160

Table 4.9: Top ranked antonym patterns for adjectives

Pattern	**LMI score**
<ANT>/NN and/CC <ANT>/NN	811.7324
<ANT>/NN and/CC <ANT>/NN ./.	715.0134
<ANT>/NN or/CC <ANT>/NN	550.9287
a/DT <ANT>/NN and/CC a/DT <ANT>/NN	528.6189
a/DT <ANT>/NN and/CC a/DT <ANT>/NN ./.	419.6441
<ANT>/NN and/CC a/DT <ANT>/NN	374.0532
<ANT>/NN and/CC a/DT <ANT>/NN ./.	366.6803
between/IN a/DT <ANT>/NN and/CC a/DT <ANT>/NN	366.2596
<ANT>/NN and/CC <ANT>/NN ,/,	352.0106
the/DT <ANT>/NN and/CC <ANT>/NN	339.3649
between/IN a/DT <ANT>/NN and/CC a/DT <ANT>/NN ./.	275.5476
<ANT>/NN or/CC <ANT>/NN ./.	217.5345
a/DT <ANT>/NN and/CC a/DT <ANT>/NN ,/,	204.3381
<ANT>/NN and/CC a/DT <ANT>/NN ,/,	177.2604
a/DT <ANT>/NN and/CC <ANT>/NN	170.7638
one/CD <ANT>/NN and/CC one/CD <ANT>/NN	166.1370
the/DT <ANT>/NN and/CC <ANT>/NN of/IN	165.3372
between/IN <ANT>/NN and/CC <ANT>/NN	158.5301
<ANT>/NN and/CC <ANT>/NN of/IN	152.4708
between/IN a/DT <ANT>/NN and/CC a/DT <ANT>/NN ,/,	145.4421

Table 4.10: Top ranked antonym patterns for nouns

Pattern	**LMI score**
<ANT>/VB or/CC <ANT>/VB	961.6148
<ANT>/VB and/CC <ANT>/VB	636.1911
<ANT>/VB or/CC <ANT>/VB ./.	347.1819
<ANT>/VBP and/CC <ANT>/VBP	295.6043
to/TO <ANT>/VB and/CC <ANT>/VB	253.0905
<ANT>/VB and/CC <ANT>/VB ./.	214.7118
<ANT>/VB or/CC <ANT>/VB the/DT	178.6803
<ANT>/VBP or/CC <ANT>/VBP	170.4069
,/, <ANT>/VB or/CC <ANT>/VB	157.9835
<ANT>/VB or/CC <ANT>/VB a/DT	145.4078

<ANT>/VB and/CC <ANT>/VB ,/,	143.7145
to/TO <ANT>/VB or/CC <ANT>/VB ./.	112.2844
<ANT>/VB or/CC <ANT>/VB ,/,	106.6906
to/TO <ANT>/VB or/CC <ANT>/VB the/DT	83.6648
can/MD <ANT>/VB or/CC <ANT>/VB	83.3056
not/RB <ANT>/VB or/CC <ANT>/VB	80.0805
<ANT>/VBP and/CC <ANT>/VBP ,/,	68.2536
<ANT>/VB and/CC <ANT>/VB the/DT	55.8227
will/MD <ANT>/VB or/CC <ANT>/VB	50.7752
can/MD <ANT>/VB and/CC <ANT>/VB	43.6558

Table 4.11: Top ranked antonym patterns for verbs

Patterns strongly associated with antonyms might not necessarily be the ones that are best at discriminating antonyms from synonyms and other word pairs. Which patterns distinguish antonyms from other word pairs is learned by the classifiers. We trained Decision Trees using a single pattern at a time as feature on the adjectival training data and plotted precision and recall of the single patterns (see Figure 4.7).

The analysis shows that [*between/IN* <ANT>/JJ *and/CC* <ANT>/JJ] and [*both/DT* <ANT>/JJ *and/CC* <ANT>/JJ] are among the top patterns concerning the trade-off between precision and recall. Shorter patterns like [<ANT>/JJ *and/CC* <ANT>/JJ] are less precise, but offer a higher recall. The analysis gives justification to Lin's intuition about patterns of incompatibility as both Lin patterns are among the patterns with the best trade-off between precision and recall.

The features of this subsection are later referred to as feature category *news105-patterns*.

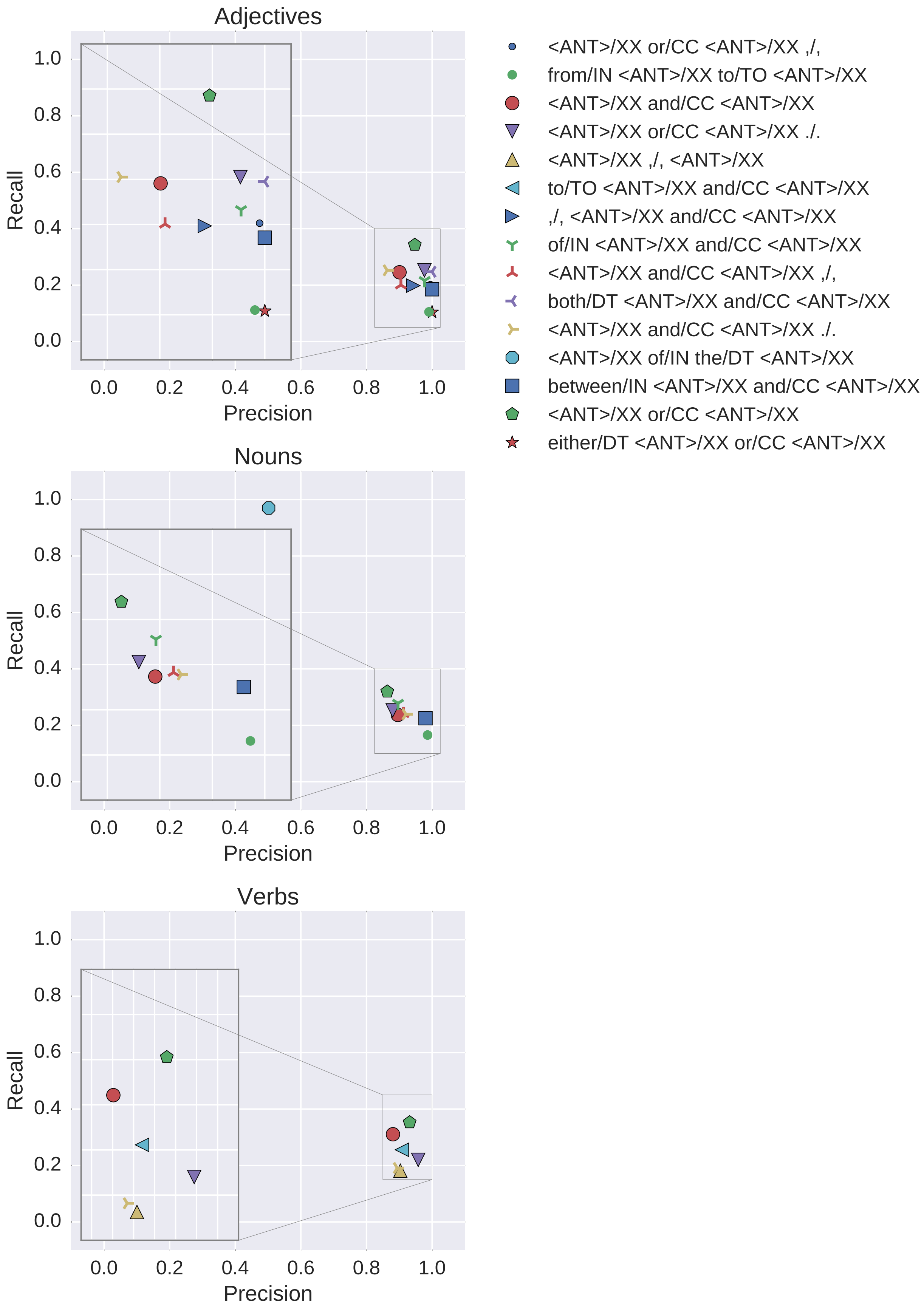


Figure 4.7: Precision versus recall of the patterns using *News105m* as corpus

In subsection 4.3.5 we showed that the recall of the Lin patterns strongly increases when relying on counts from the Google N-gram corpus instead of counts from *News105m*. So, we ported our pattern-based approach to this corpus. We did not mine the Google N-gram corpus for antonym patterns again. Instead, we took the antonym patterns, we mined from *News105m* and discarded the part-of-speech information. Also, we filtered out all patterns longer than five tokens, because there are no N-gram counts available in the Google corpus for these.

Precision vs. recall of the patterns

Again, we trained each a Logistic Regression classifiers for the single pattern features to compare the importance of the patterns for the antonym classification. Results are shown in Figure 4.8.

For verbs, adjectives and nouns the patterns [<X> *or* <Y>] and [<X> *and* <Y>] are in the mid range of recall and precision. For all of the part-of-speech categories the patterns [*either* <X> *or* <Y>] and [*both* <X> *and* <Y>] have a high precision but their recall is not high. For adjectives and verbs the pattern [*between* <X> *and* <Y>] scores as well as the patterns [*either* <X> *or* <Y>] and [*both* <X> *and* <Y>]. But for verbs the pattern is less precise. [*from* <X> *to* <Y>] is only precise for adjectives, but does not work as well with nouns and verbs. Thus, from the two Lin patterns [*either* <X> *or* <Y>] and [*from* <X> *to* <Y>] only the former one scores well for all part-of-speech categories. [*both* <X> *and* <Y>] scores as well as [*either* <X> *or* <Y>] and also [*both* <X> *and* <Y>] should be considered to rule out contrasting pairs in a way Lin proposed.

Comparing the results using Google's N-gram corpus and the results using *News105m* two observations can be made. Mostly the same patterns offer the best recall-precision trade-off. And the recall of the patterns increases strongly using the N-gram corpus. Whereas the best pattern has a recall of about 0.4 with the smaller corpus, with the large-scale web counts the best pattern has a recall of more than 0.7.

Does it make sense to separate the part-of-speech categories?

Figure 4.8 indicates that the patterns with the best recall-precision trade-off for the different part-of-speech categories do not differ too much. Thus, we heuristically checked whether the separation into the different categories makes sense. We trained a logistic regression model each for the different part-of-speech categories and examined the patterns with the lowest coefficients (see Table 4.12). Note that the patterns with the lowest coefficients are the ones which indicate antonymy since in the mathematical model the class antonymy is denoted with one and the class non-antonymy is denoted with two.

Adjectives		**Nouns**		**Verbs**	
Coefficient	Pattern	Coefficient	Pattern	Coefficient	Pattern
-1.27e+00	both X and X	-1.41e+00	both X and Y	-1.03e+00	X to a Y

Adjectives		Nouns		Verbs	
Coefficient	Pattern	Coefficient	Pattern	Coefficient	Pattern
-1.10e+00	X than Y	-1.39e+00	between X and Y	-9.48e-01	X or Y ,
-1.03e+00	either X or Y	-1.06e+00	the X or Y	-8.43e-01	one X and one Y
-9.73e-01	X , not the Y	-9.08e-01	X , not Y	-8.05e-01	X , not Y
-9.09e-01	from X to Y	-8.36e-01	the X or Y of	-7.75e-01	X or Y
-8.93e-01	the X or the Y	-7.77e-01	at the X and Y	-6.85e-01	X as well as Y
-8.72e-01	X as well as Y	-6.85e-01	X to a Y	-6.66e-01	the X or the Y
-8.16e-01	for X and Y	-6.60e-01	, X or Y ,	-6.32e-01	, X or Y
-7.92e-01	between X and Y	-6.54e-01	X news and Y	-6.30e-01	either X or Y
-7.89e-01	’s X and Y	-6.03e-01	, X and Y ,	-6.24e-01	is X and Y
-7.57e-01	all X and Y	-5.41e-01	X and Y	-5.84e-01	are X and Y
-7.53e-01	X or Y	-5.38e-01	X as well as Y	-5.54e-01	not X and Y
-6.13e-01	, X and Y	-5.23e-01	of X and Y	-5.47e-01	to X or Y the
-5.92e-01	, X and Y ,	-5.00e-01	for X or Y	-5.33e-01	of X , Y
-5.71e-01	of X and Y	-4.98e-01	a X , a Y	-5.25e-01	both X and Y
-5.70e-01	about X and Y	-4.76e-01	X round Y	-5.07e-01	an X , Y
-5.49e-01	The X and Y	-4.61e-01	like X and Y	-5.06e-01	the X and Y
-5.44e-01	X nor Y	-4.60e-01	from X and Y	-4.50e-01	neither X nor Y
-5.41e-01	X to that Y	-4.56e-01	from X to Y .	-4.22e-01	by X and Y
-5.24e-01	X of the Y for	-4.50e-01	a X or a Y	-4.19e-01	in X and Y
⋮	⋮	⋮	⋮	⋮	⋮
4.97e-01	X , Y	5.73e-01	X quarter Y	5.05e-01	X nor Y
5.26e-01	the X to the Y	5.78e-01	to X and Y	5.19e-01	in the X , Y
5.40e-01	from X and Y	5.87e-01	from X to Y and	5.33e-01	X for a Y
5.83e-01	on X and Y	5.98e-01	X of the Y	5.42e-01	X against Y
6.56e-01	with X and Y	6.01e-01	a X and Y .	5.83e-01	from X to Y and
6.66e-01	– X , Y	6.47e-01	the X , Y	5.94e-01	X to Y .
6.87e-01	X and Y ,	7.10e-01	X to Y .	7.09e-01	of X and Y
7.59e-01	from X to Y and	7.45e-01	about X and Y	7.31e-01	of X or Y
7.72e-01	X on a Y	8.83e-01	X or a Y	9.12e-01	as X and Y
1.03e+00	by X and Y	8.95e-01	X and the Y ,	9.55e-01	– X , Y

Table 4.12: The coefficients of logistic regression models for the different part-of-speech categories from lowest to highest. Low coefficients indicate antonymy.

We computed the overlap of the top n patterns indicating antonymy according to the coefficients of logistic regression models for the different part-of-speech categories. Results are shown in Table 4.13.

For the top 20 patterns the overlap between the different categories is about 20%, for the top 50 patterns 30% and the top 100 patterns 40%. These numbers rather hint that there are notable differences between the part-of-speech categories. Using only the features from the Google corpus

n	Adjectives-Nouns	Adjectives-Verbs	Noun-Verbs
20	5	5	4
50	15	19	15
100	42	38	41

Table 4.13: Overlap of the top n patterns indicating antonymy according to the coefficients of logistic regression models for the different part-of-speech categories.

we obtained F-scores for a 10-fold cross-validation with random forest models of 0.8371 for the adjective pairs, 0.8049 for the noun pairs and 0.7401 for the verb pairs. Combining the data we obtained an F-score of 0.7691. This score is about 0.025 lower than the unweighted mean of the three single F-scores, which is 0.7940, and about 0.043 lower than the mean F-score of the part-of-speech categories weighted by training size, which is 0.8117. Those scores again indicate, that it makes sense to train different classifiers for the part-of-speech categories since normally one would expect an even better F-scores for the combination of the training sets, because there is considerably more data available for training. And more training data most often leads to better results.

Pruning

Furthermore, we checked whether pruning to a certain extent leads to better results, finding no evidence that pruning is helpful. Table 4.14 lists the F-scores keeping only the top n patterns determined by the total association score with antonym patterns from *WordNetPairs*. The best pattern alone achieves a score of about 0.69 for the noun data. The results get continuously better adding more patterns. This holds also for the adjective and verb data.

n	F-score adjectives	F-score nouns	F-score verbs
1	0.7943	0.6939	0.6744
5	0.7933	0.6883	0.6809
10	0.7994	0.7020	0.6744
50	0.7974	0.7058	0.6841
100	0.8048	0.7268	0.6922
200	0.8020	0.7423	0.7215
300	0.8043	0.7569	0.7051
400	0.8047	0.7476	0.7134
500	0.8098	0.7585	0.7135

Table 4.14: F-score for the antonyms vs all task on the noun pairs from *WordnetPairs*

The features of this subsection are later referred to as feature category *web1t-patterns*.

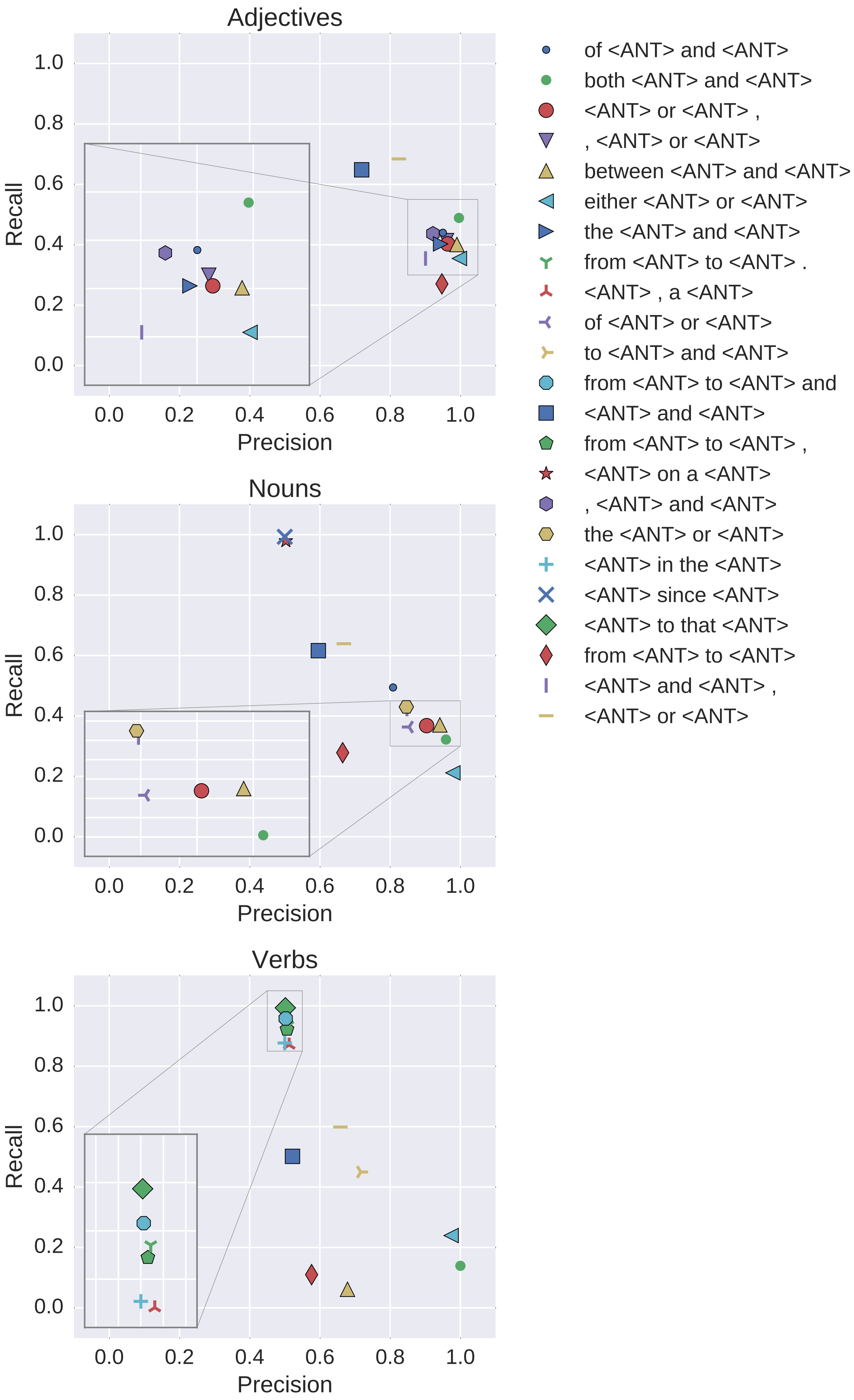


Figure 4.8: Precision versus recall of the patterns using the Google N-gram corpus

Having gathered enough training data and having extracted many features from different feature categories, the next step is to find a suitable combination of features and to evaluate which classifier to choose and to find the best hyperparameter setting for the chosen classifier.

First, we standardized our data in the way. We subtracted from each feature its mean value to make each feature have zero-mean and we divided each feature by its standard deviation to make each feature have unit-variance. This standardization is useful or even necessary for some of the learning algorithms we tested including SVMs with different kernels and Logistic Regression. Other learning algorithms do not depend on feature normalization, e.g. Decision Trees and Random Forests. But these algorithms do not produce worse results if the features are indeed normalized.

We used 10-fold cross-validation to obtain mean F-scores on the WordNet adjectival training data for several learning algorithms[2] (Henceforth, every time we refer to a cross-validation unless specified otherwise a 10-fold cross-validation was conducted.) Table 4.15 shows the obtained scores. At this point, we used the standard hyperparameter settings of their respective implementations.

	F-score adjectives	**F-score nouns**	**F-score verbs**
Nearest Neighbors	0.8793	0.8496	0.7554
Linear SVM	0.9133	0.8780	0.7826
RBF SVM	0.7883	0.6477	0.6283
Decision Tree	0.9164	0.8559	0.7775
Random Forest	0.9274	0.8909	0.8411
Naive Bayes	0.7753	0.8255	0.7580
Logistic Regression	0.8868	0.8550	0.7744

Table 4.15: Comparing different classifiers on the training data

For all part-of-speech categories a random forest fits the data best, which is why we chose random forest models for further evaluation of our approach. The worst results were obtained with a radial basis function SVM and Naive Bayes. As the results with a linear SVM show, a radial basis function is the wrong kernel for our data. Naive Bayes does not work well for our data since for example features from feature categories like *web1t-patterns* and *news105-patterns* are not independent, which violates the basic assumption of Naive Bayes.

The Random Forest algorithm offers some hyperparameters to tune including the number of Decision Trees to train and some tree pruning factors such as maximum tree depth. We ran a grid search testing different hyperparameter settings finding no considerable performance decline or improvement of a specific setting. Hence, we settled for the standard parameters. The following experiments are conducted with this algorithm configuration.

Two experiments were carried out to find the best feature combination. We performed cross-validations using one feature category at a time and we conducted ablation tests. Results of the first experiment are shown in Table 4.16.

[2] We used implementations of the learning algorithms from Scikit Learn: `http://scikit-learn.org/stable/`

For all of the feature categories the precision is higher than the recall. For example the Lin patterns produce precision scores of above 98% on all part-of-speech categories but low recall scores. Some feature categories like *jb-sim*, *word-embed* and *para-cooc* score low on their own, but we later show that they are helpful in combination with the pattern features.

The best scores yields feature category *web1t-patterns* with about 84% for the adjective data, 80% for the noun data and 74% for the verb data.

It is interesting that *news105-context*, which is calculated on the same corpus as *news105-patterns* and comprises mostly the same information, produces better F-scores on the adjective and the verb data. *news105-context* provides association scores of the pairs with the words of the top ranked patterns regarding a sentence window, for instance association scores of “both” and “between” and “and” with the pairs. This information should conceptually be less precise than the pair-pattern association of *news105-patterns*. But, only for the noun data we obtained the expected result. This can be interpreted as evidence that dependency patterns could improve the results. The pairs might sometimes not occur in the exact patterns but slightly altered infrequent versions of the patterns. This is captured by *news105-context* but not by *news105-patterns*.

	Adjectives			**Nouns**			**Verbs**		
	P	R	F	P	R	F	P	R	F
sen-cooc	0.8219	0.6582	0.7478	0.7044	0.4893	0.6174	0.7867	0.5601	0.6952
news105-context	0.8410	0.7144	0.7817	0.7636	0.6279	0.7045	0.8143	0.6698	0.7554
news105-lin	0.9922	0.2445	0.5583	0.9846	0.2598	0.5675	0.9875	0.1683	0.4951
jb-sim	0.8031	0.3597	0.5935	0.7097	0.2805	0.5343	0.7840	0.3458	0.5632
morpho	0.8953	0.6643	0.7861	0.8919	0.6805	0.7885	0.8440	0.5272	0.7021
word-embed	0.6393	0.5923	0.6133	0.6359	0.6159	0.6232	0.5876	0.5306	0.5582
news105-patterns	0.9332	0.5432	0.7402	0.8789	0.5999	0.7498	0.9021	0.5247	0.7204
para-cooc	0.8262	0.3628	0.6093	0.7281	0.2717	0.5327	0.7577	0.2814	0.5477
web1t-patterns	0.9107	0.7556	0.8371	0.8557	0.7430	0.8049	0.8193	0.6276	0.7401

Table 4.16: Cross-validations on *WordNetPairs* with a single feature category each

The results of the ablation test can be inspected in Table 4.17. Discarding morphological clues leads to the heaviest decrease in F-score. The score for e.g. adjectives drops from about 93% by 7% to 86%. Plenty of the antonym pairs in WordNet match one of the morphological patterns in subsection 4.3.1, which explains this drop. Excluding the features from *web1t-patterns* results in another significant F-score reduction. This, again, makes sense, because counting only on pair-pattern association scores using the Google N-gram corpus for instance yields an F-score of about 82% for the adjective data.

Excluding some feature categories produces slightly higher scores than using all features. The differences are all below 0.5%. We decided to just use all the features for our final model because for all feature categories the values had already been computed. In case computation time is of importance, it might make sense to exclude some feature categories and/or to use more elaborate feature selection methods.

	Adjectives			Nouns			Verbs		
	P	R	F	P	R	F	P	R	F
sen-cooc	0.9378	0.9309	0.9334	0.8955	0.9060	0.8956	0.8604	0.8351	0.8458
news105-context	0.9365	0.9189	0.9269	0.8969	0.9059	0.8979	0.8661	0.7961	0.8329
news105-lin	0.9380	0.9289	0.9324	0.8884	0.9036	0.8909	0.8608	0.8318	0.8460
jb-sim	0.9451	0.9259	0.9349	0.8890	0.9129	0.8956	0.8479	0.8415	0.8428
morpho	0.8458	0.8808	0.8541	0.8212	0.8300	0.8206	0.7832	0.8060	0.7879
word-embed	0.9367	0.9309	0.9329	0.8946	0.9082	0.8966	0.8381	0.8220	0.8281
news105-patterns	0.9355	0.9269	0.9298	0.9003	0.9013	0.8966	0.8473	0.8446	0.8427
para-cooc	0.9441	0.9269	0.9349	0.8858	0.9061	0.8910	0.8470	0.8352	0.8395
web1t-patterns	0.9252	0.9078	0.9153	0.8838	0.8854	0.8781	0.8036	0.8254	0.8045
all	0.9349	0.9269	0.9299	0.8887	0.9013	0.8897	0.8555	0.8349	0.8445

Table 4.17: Ablation test on *WordNetPairs*. The last row contains the results with all feature categories. The other rows show the results without the feature category given in the first column.

Finally, we evaluated the chosen setting on all of the WordNet data. We performed our experiments for the adjective, noun and verb data each in two ways. On the one hand, we trained binary classifiers for the antonymy vs synonymy task, on the other hand, we trained binary classifiers for the task of discriminating antonymous word pairs from word pairs of any kind. It does not seem that there is a difference in difficulty between those two tasks since the F-scores at most differ slightly. Regarding part-of-speech tag categories identifying antonymous verbs seems to be the most difficult. The F-scores drop about 10% compared to the adjectival data.

Cross-validation on WordNetPairs	**F-score**
Adjectives: antonymy vs synonymy	0.93
Adjectives: antonymy vs all	0.93
Nouns: antonymy vs synonymy	0.90
Nouns: antonymy vs all	0.90
Verbs: antonymy vs synonymy	0.82
Verbs: antonymy vs all	0.84

Table 4.18: Cross-validating on WordNet data

To conduct an error analysis on the WordNet data we obtained predictions for all the labeled instances in the following way. We performed a 3-fold cross-validation, collected the predictions for the test sets and combined them.

Adjectives

We attained a precision of 0.92, a recall of 0.92 and a F-score of 0.92 on the adjective pairs from WordNet. There were 85 false positives and 80 false negatives (see Table 4.19).

		Predicted class:	
		antonyms	non-antonyms
True class:	antonyms	918	80
	non-antonyms	85	913

Table 4.19: Confusion matrix for adjective pairs

The association scores of the pairs with the antonym patterns in the Google N-gram corpus make up lots of the top features of our Random Forest Model (Feature importance was measured as described in [Breiman et al., 1984]). Thus, we examined the scores, looking for commonalities of especially the true positives and the true negatives. We determined how often the association scores are on average positive for the top 50 features.

	True positives	**True negatives**	**False positives**	**False negatives**
Mean	12.1885	0.7338	5.8941	2.3125
Median	8	0	4	0
Standard deviation	11.5734	1.8703	7.1331	4.5801

Table 4.20: Mean/median/standard deviation of the number of lmi scores bigger than zero in the top 50 features from feature category *web1t-pattern* for the adjective data from WordNet

Results are shown in Table 4.20. For pairs in the true positives the mean number of positive association scores with the top 50 *web1t-pattern* features is about 12.1885. For pairs in the true negatives it is 0.7338. This indicates that the number of positive scores correlates positively with predicting the class antonym. We see that false positives are often also associated with many of the top patterns. The mean is 5.8941. We also included the median because the mean might be strongly influenced by outliers. In this case the median is 4, which is lower than the mean but still shows that the false positive pairs often co-occur with some of our antonym patterns. We counted for the top patterns how often they have positive association scores with false positive pairs. See Table 4.21 for an excerpt of the results.

False positive pairs very often occur with the patterns [<X> *or* <Y>], [<X> *and* <Y>] and [<X> , <Y>]. We earlier showed that those patterns offer a high recall for antonym identification, but not a high precision. In fact, as we see now, they seem to be responsible for lots of the false positives. More precise patterns like [*between* <X> *and* <Y>] only seldomly co-occur with false positives.

Pattern	Number of positive association scores with false positives
[<X> *or* <Y>]	44
[<X> *and* <Y>]	42
[<X> , <Y>]	42
[*both* <X> *and* <Y>]	0
[*between* <X> *and* <Y>]	2
[*either* <X> *or* <Y>]	1

Table 4.21: Patterns that are often associated with pairs from the set of the false positives

Another source of error are word pairs that are tagged as synonyms in WordNet and are very close in writing like „blond blonde“, „licenced licensed“, „economic economical“ or „polemic polemical“. Those pairs produce high values for some of the features in feature category *morpho*, which are also among the top features of our model.

An interesting false positive pair is „experimental observational“. The two words are tagged as synonymous in WordNet. But at least in the medicinal domain they are often used contrastingly. There are experimental studies and observational studies, which are two different things. This is one rare case where antonymy seems to be domain-dependent.

Nouns

We attained a precision of 0.87, a recall of 0.87 and a F-score of 0.87 on the noun pairs from WordNet. There were 64 false positives and 46 false negatives (see Table 4.22).

		Predicted class:	
		antonyms	non-antonyms
True class:	antonyms	389	46
	non-antonyms	64	371

Table 4.22: Confusion matrix for noun pairs

Most of the false positive pairs are synonyms (36), 14 are hyponyms (see also Table 4.23).

Relation	Number of false positives
synonyms	36
hyponyms	14
meronyms	9
cohyponyms	5
unrelated	0

Table 4.23: True relation of the false positive pairs

Again, We counted how often the pairs are positively associated with the top 50 patterns. Results can be inspected in Table 4.24.

	True positives	True negatives	False positives	False negatives
Mean	11.9434	2.8355	8.4688	6.0652
Median	8	0	4	3
Standard deviation	6.4567	1.8703	10.1543	8.5645

Table 4.24: Mean/median/standard deviation of the number of lmi scores bigger than zero in the top 50 features from feature category *web1t-pattern* for the noun data from WordNet

False positive pairs on average were positively associated with the top pattern about 8.5 times, which is about 6 times more than true negative pairs. Again, the patterns [<X> *or* <Y>], [<X> *and* <Y>] and [<X> *,* <Y>] were often positively associated with false negative pairs (see Table 4.25).

Pattern	Number of positive association scores with false positives
[<X> *or* <Y>]	29
[<X> *and* <Y>]	35
[<X> *,* <Y>]	32
[*both* <X> *and* <Y>]	5
[*between* <X> *and* <Y>]	4
[*either* <X> *or* <Y>]	2

Table 4.25: Patterns that are often associated with pairs from the set of the false positives

We observed that a slightly bigger ratio of false positive noun pairs than adjective pairs were associated with high precision patterns like [*both* <X> *and* <Y>]. and [*between* <X> *and* <Y>]. Among the pairs associated with the later pattern is „hearing listening“, which is related by synonymy according to WordNet. In WordNet the sense „the act of hearing attentively“ is linked by synonymy with listening. Our classifier might have found hearing in a sense of „the act of hearing inattentively“ to be antonymous to listening. Other pairs that occur together with high precision patterns are the co-hyponyms „kicker punter“ and „fullback halfback“, which are football positions.

An interesting false positive pair is „ job taks“. In WordNet the pair is annotated as synonymous. But, the pair has a high association score with the pattern [*from* <X> *to* <Y>], which is very uncommon for synonyms. In fact, job and task are, in the software domain, often used for different things. Tasks are often the steps/parts of a job or in other definitions jobs are used to reserve the resources required by tasks.

Of the 46 false negative pairs 14 (about 30%) cannot be found in 3-grams in the Google N-gram corpus and 33 (about 70%) do not co-occur in a sentence in *News105m*. Of all antonym pairs 10% cannot be found in 3-grams in the Google corpus and 38% not in the news corpus. Examples of false negative pairs that do not co-occur in a 3-gram window are for example „difference sameness“ and „coldness hotness“.

Verbs

We attained a precision of 0.82, a recall of 0.82 and a F-score of 0.82 on the verb pairs from WordNet. There were 63 false positives and 49 false negatives (see Table 4.26).

		Predicted class:	
		antonyms	non-antonyms
True class:	antonyms	260	49
	non-antonyms	63	247

Table 4.26: Confusion matrix for verb pairs

An interesting false positive pair is „call visit“. In *WordNetPairs* the pair is annotated as a random pair. The pair is highly associated with patterns like [*either* <X> *or* <Y>] and [*from* <X> *to* <Y>] that commonly indicate antonymy. We hypothesize that the contrasting options ”just call” or ”really go visiting” make our classifiers see the pair as antonyms. Apart from that, the errors for verb pairs are similar to the ones for adjectives and nouns. Again, pairs similar in writing like „crystallise crystallize“ produce several false positive pairs. Also, high association scores with the patterns [<X> *or* <Y>], [<X> *and* <Y>] and [<X> *,* <Y>] often lead to false positive predictions.

Generally we observed four major error categories:

- High similarity in writing produces false positive pairs.
- High association with the following patterns produces false positive predictions:
  - [<X> *or* <Y>]
  - [<X> *and* <Y>]
  - [<X> *,* <Y>]
- Domain-specific or situation-specific contrast leads to false positives.
- Antonyms that do not co-occur in sentences are not recognized.

5 Comparative Evaluation

It is customary to evaluate a NLP system or algorithm intrinsically and extrinsically [Baroni and Lenci, 2011, Jones and Galliers, 1995].

Intrinsic evaluation is about testing the system in itself, e.g. testing a classifier on a test set.

Extrinsic evaluation is about testing the system on a secondary task. Biemann and Riedl [2013], for instance, evaluated the quality of their DT extrinsically on the task of POS-tagging. A common source of errors assigning POS-tags are unknown words, words which did not occur in the training data of the tagger. A DT can help to mitigate this problem. Encountering an unknown word, the word can often be substituted with the help of the DT with a distributionally similar word which occurred in the training data of the tagger. Biemann and Riedl [2013] showed in this task-oriented evaluation, that their DT can improve the accuracy of part-of-speech tagging.

We already evaluated our classifiers intrinsically in the last chapter performing cross-validations on our own data sets. Since this is not a sufficient intrinsic evaluation we tested our classifiers on some more test sets from literature. These experiments are described in section 5.1.

5.1 Intrinsic Evaluation

To be able to compare our classifiers for English word pairs to results from literature, we used two data sets. The first one (*AntSynLin*) from Lin et al. [2003] contains 80 antonymous noun pairs and 80 synonymous noun pairs. The second one (*AntSynTurney*) from Turney [2008] consists of 137 antonymous and synonymous pairs of mostly nouns and adjectives, but also verbs.

Table 5.1 provides an overview of the data sets used for intrinsic evaluation.

	ANT	SYN	HYP	COHYP	MER	UNR
WordNetPairs						
NOUN	466	94	93	93	93	93
ADJ	999	450				449
VERB	310	155				155
AntSynLin						
NOUN	80	80				
AntSynTurney						
NOUN	15	7				
ADJ	43	26				

	ANT	SYN	HYP	COHYP	MER	UNR
VERB	31	14				

Table 5.1: Data Sets

AntSynLin

On *AntSynLin* we reach a F-score of 81%. Lin et al. [2003] report a score of about 90%. Lin et al. [2003] worked with counts from search engines. Most probably the difference in F-score stems from Lin et al. [2003] accessing a larger corpus, albeit in an irreproducible manner. Our classifier produced 65 true positives, 63 true negatives, 16 false positives and 15 false negatives. Among the false positives were for instance the pairs „art craft“ and „prison college“. Both are synonym pairs according to *AntSynLin*. „art craft“ can be found in several of the patterns we found to indicate antonymy in our WordNet analysis like [*both* <X> *and* <Y>] and [*between* <X> *and* <Y>]. Intuitively, we would think of art and craft as something contrasting, with art being nonfunctional and craft being functional. College is a slang term for prison and thus the pair „college prison“ is a synonym pair in *AntSynLin*. Most probable the high association of the pair with [*from* <X> *to* <Y>] which we would rather attribute to college in an educational sense is what produces this false positive pair.

Among the false negative pairs are „average maximum“ and „average minimum“. Whereas our classifiers recognize „maximum minimum“ as an antonym pair they fail to detect the incompatibility of the middle „average“ and the poles of the scale ranging from minimum to maximum. Another false negative pair is „exaggeration understatement“, which only rarely, 370 times, co-occurs in 3-grams in the Google N-gram corpus. Other false negative pairs like „exodus influx“ never co-occur in any N-gram of the Google corpus.

AntSynTurney

On *AntSynTurney* we reach a F-score of 77% which seems to be slightly better than the pure pattern-based approach of Turney [2008] which produced a score of 75%.

Among the false positive pairs are for instance pairs like „generic general“ that are close in writing and pairs that have high association scores with [<X> *and* <Y>] and [<X> *or* <Y>] like „university college“. Among the false negatives are pairs like „permit deny“ that never co-occur in a N-gram in the Google Corpus.

5.2 Extrinsic Evaluation

We use the GRE data set for a task-oriented evaluation of our classifiers for the English language. This data sets consists of questions containing a target word and five choices. The task is to determine the choice as response which is closest to being an antonym of the target word. Table 5.2 shows an example question from the GRE data set.

Target	Choices	Response
solvent	enigmatic, bankrupt, fiducial, puzzling, gilded	bankrupt
penchant	distance, imminence, dislike, attitude, void	dislike
paucity	pouch, peace, quickness, abundance, nuisance	abundance
lithe	stiff, limpid, facetious, insipid, vast	stiff
lurid	dull, duplicate, heavy, grotesque, intelligent	dull
malevolent	kindly, vacuous, ambivalent, volatile, primitive	kindly
manifest	limited, obscure, faulty, varied, vital	obscure
impecunious	affluent, afflicted, affectionate, affable, afraid	affluent
inane	passive, wise, intoxicated, mellow, silent	wise
maniacal	demoniac, saturated, sane, sanitary, handcuffed	sane

Table 5.2: Data sample from the GRE data set.

The GRE data comes in two parts. A development set containing 950 questions and the test set containing 790 questions. We did not use the development set, we only tested our classifiers on the test set. Table 5.3 provides previous results on the GRE test set. We considered two different baselines for the systems. Guessing the answer would result in a F-score of 0.2. A system which would look up the target and choice pairs in WordNet and take a guess if none of the word pairs can be found in WordNet would come up with a score of 0.23.

	F-score
Baseline	0.2
Wordnet lookup baseline	0.23
Lin et al. [2003]	0.24
Mohammad et al. [2013]	0.69
Ono et al. [2015]	0.88

Table 5.3: Results on the GRE test set

The achieved F-scores range from 0.24 to 0.88. 0.24 is achieved by a re-implementation of the method of Lin et al. [2003] from Mohammad et al. [2013]. Mohammad et al. [2013] answer the questions with an antonym list generated from different lexical resources using several heuristics. They reach a F-score of 0.69. The best score, 0.88, is reported by Ono et al. [2015] who trained specialized word embeddings using antonymy and synonymy information from lexical resources.

We relied on the confidence values of our classifiers' predictions to answer the questions. We fed our classifier with all target/choice pairs. Then, we picked the choice as our answer which resulted in the highest confidence value for the class antonym. Proceeding this way, our system yielded a F-score of about 0.32. We inspected the confidence values for the class antonym of the target/choice pairs to look for an answer to the question why our system performs rather bad in comparison. For most of the questions none of the choices is considered contrasting to the target by our classifiers. For most of the questions even none of the confidence values for antonymy is higher than 0.1. See Table 5.4 for sample results for some questions.

target	**choices**					**real answer**
incipient	exuberant	full-bodied	explicit	plentiful	full-blown	full-blown
	0.0	0.0	0.0	0.0	0.0	
veneration	derision	blame	avoidance	ostracism	defiance	derision
	0.08	0.0	0.0	0.01	0.03	
testiness	devotion	patience	methodicalness	caution	discretion	patience
	0.04	0.04	0.06	0.04	0.0	
transparent	indelicate	neutral	opaque	somber	tangible	opaque
	0.02	0.4	0.9	0.02	0.09	
asset	duty	qualification	denial	liability	instability	liability
	0.38	0.06	0.11	0.82	0.08	

Table 5.4: Sample results for GRE. The choices are mapped to the confidence score of the classifiers for predicting the class antonymy.

We proceeded to compute a F-score for the results for only a subset of the questions. We only regarded the questions where at least one of the confidence scores was 0.5 our higher. For this subset of the GRE test set we reached a F-score of about 0.65. But, only 68 of 790 could be answered this way. The low F-score achieved by the method of Lin et al. [2003], who used two patterns which are also in our automatically mined patterns, hints into the right direction. Only 139, about 17.6%, of the 790 most contrasting choices co-occur with their respective target word into 3-grams in the Google N-gram corpus. This data sparseness makes it impossible to reach a high F-score with a pattern-based approach and the Google corpus.

6 Conclusion and Future Work

6.1 Conclusion

We found that the predominantly pattern-based approach works well on established and frequent antonyms from WordNet. Results ranged from F-scores of about 0.82 to 0.93 for the antonymy versus synonymy task and from 0.84 to 0.93 for the task of classifying random word pairs as being antonyms or not. Regarding different part-of-speech tag categories, identifying antonymy seems to be most difficult for verbs.

An analysis on which patterns are the most useful for identifying antonymy gave experimental evidence for the intuition of Lin et al. [2003] who used the patterns [*either* <X> *or* <Y>] and [*from* <X> *to* <Y>] to identify antonyms. Our analysis showed that [*both* <X> *and* <Y>] and [*between* <X> *and* <Y>] are as useful as the Lin patterns.

We showed that more data improves the pattern based approach a lot. With a 105 million sentences corpus we reached a F-score of 0.72 for the antonymy versus all task on our WordNet adjectival data. Using the Google N-gram corpus we reached a F-score of 0.82.

Our results on antonymy versus synonymy data sets from Lin et al. [2003] and Turney [2008] were in line with the literature; we reached F-scores of 0.81 and 0.77 for these data sets, versus 0.9 and 0.75 reported in literature.

A task-oriented evaluation on the GRE data set uncovered the weaknesses of our approach. Only for about a tenth of the questions our classifier found one of the answer choices to be antonymous with a confidence higher than 50%. These results can be explained with data sparseness. Most of the contrasting pairs in the GRE data do not co-occur at all in the N-grams from the Google N-gram corpus.

6.2 Future Work

We focused a lot on the antonymy versus synonymy task which was examined among others by Lin et al. [2003], Turney [2008] and Santus et al. [2014]. The results on our data from WordNet hint that antonymy can be distinguished from other relations like hyponymy or co-hyponymy at least equally well as from synonymy. However, we did not examine this in depth, posing this as possible future work.

One strength of the pattern based approach is that a relatively small training set is sufficient. Hence, it is possible to port the approach to other languages without much effort. The Google N-Gram corpus is also available in the same format in for example German, Spanish and French. Thus, it would be interesting to concentrate on the pattern features from this N-gram corpus and port the approach to the other languages available in the corpus.

Furthermore, the approach could be included into or generalized for ontology induction from text. It might be interesting to evaluate if it is possible to create lexical resources such as WordNet without relying on manual annotation.

Number	Tag	Description
1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential *there*
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	*to*
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun

Number	Tag	Description
36.	WRB	Wh-adverb

Table 7.1: Alphabetical list of part-of-speech tags used in the Penn Treebank Project

List of Figures

List of Tables

Marco Baroni and Alessandro Lenci. How we BLESSed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on Geometrical Models of Natural Language Semantics*, pages 1–10. Association for Computational Linguistics, 2011. (Cited on page 57).

Chris Biemann and Martin Riedl. Text: Now in 2D! a framework for lexical expansion with contextual similarity. *Journal of Language Modelling*, 1(1):55–95, 2013. (Cited on pages 16, 17, 18 and 57).

Stefan Bordag. A comparison of co-occurrence and similarity measures as simulations of context. In *Proceedings of the 9th International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing'08, pages 52–63, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 3-540-78134-X, 978-3-540-78134-9. URL `http://dl.acm.org/citation.cfm?id=1787578.1787584`. (Cited on page 14).

Thorsten Brants and Alex Franz. Web 1T 5-gram version 1. 2006. (Cited on pages 32 and 39).

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324. URL `http://dx.doi.org/10.1023/A:1010933404324`. (Cited on page 24).

Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984. ISBN 0-534-98053-8. (Cited on page 53).

Walter G. Charles and George A. Miller. Contexts of antonymous adjectives. *Applied Psycholinguistics*, 10:357–375, 9 1989. ISSN 1469-1817. doi: 10.1017/S0142716400008675. URL `http://journals.cambridge.org/article_S0142716400008675`. (Cited on page 35).

Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Comput. Linguist.*, 16(1):22–29, March 1990. ISSN 0891-2017. URL `http://dl.acm.org/citation.cfm?id=89086.89095`. (Cited on page 13).

D. Alan Cruse. *Lexical semantics*. Cambridge University Press, 1986. (Cited on page 10).

James R. Curran. Ensemble methods for automatic thesaurus extraction. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 222–229, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1118693.1118722. URL `http://dx.doi.org/10.3115/1118693.1118722`. (Cited on page 16).

Marie-Catherine De Marneffe and Christopher D. Manning. Stanford typed dependencies manual. Technical report, 2008. (Cited on page 11).

Ferdinand de Saussure. *Course in General Linguistics*. Duckworth, London, [1916] 1983. (trans. Roy Harris). (Cited on pages 15 and 16).

Pedro Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55(10): 78–87, October 2012. ISSN 0001-0782. doi: 10.1145/2347736.2347755. URL `http://doi.acm.org/10.1145/2347736.2347755`. (Cited on page 20).

Ted Dunning. Accurate methods for the statistics of surprise and coincidence. *Comput. Linguist.*, 19 (1):61–74, March 1993. ISSN 0891-2017. URL `http://dl.acm.org/citation.cfm?id=972450.972454`. (Cited on page 14).

Stefan Evert. *The statistics of word cooccurrences: word pairs and collocations.* PhD thesis, Universität Stuttgart, Holzgartenstr. 16, 70174 Stuttgart, 2005. URL `http://elib.uni-stuttgart.de/opus/volltexte/2005/2371`. (Cited on page 13).

Christiane Fellbaum. Co-occurrence and antonymy. *International journal of lexicography*, 8(4): 281–303, 1995. (Cited on page 36).

J. Firth. *A synopsis of linguistic theory 1930-1955.* Studies in Linguistic Analysis, Philological. Longman, 1957. (Cited on page 15).

W. Nelson Francis and Henry Kucera. Brown corpus manual, manual of information to accompany a standard corpus of present-day edited American English, for use with digital computers. *Brown University*, 1979. (Cited on page 36).

Birgit Hamp and Helmut Feldweg. Germanet - a lexical-semantic net for German. In *In Proceedings of ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, pages 9–15, 1997. (Cited on pages 8 and 28).

Zellig S. Harris. Distributional structure. *WORD*, 10(2-3):146–162, 1954. doi: 10.1080/00437956.1954.11659520. URL `http://dx.doi.org/10.1080/00437956.1954.11659520`. (Cited on pages 9 and 15).

Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*, COLING '92, pages 539–545, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics. doi: 10.3115/992133.992154. URL `http://dx.doi.org/10.3115/992133.992154`. (Cited on pages 30 and 66).

Karen Sparck Jones and Julia R Galliers. *Evaluating natural language processing systems: An analysis and review*, volume 1083. Springer Science & Business Media, 1995. (Cited on page 57).

Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000. ISBN 0130950696. (Cited on page 10).

John S. Justeson and Slava M. Katz. Co-occurrences of antonymous adjectives and their contexts. *Computational linguistics*, 17(1):1–19, 1991. (Cited on page 35).

Maire Weir Kay. *Webster's Collegiate Thesaurus.* Merriam-Webster, 1988. (Cited on page 27).

Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966. (Cited on page 34).

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. Do supervised distributional methods really learn lexical inference relations? In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 970–976, 2015. URL `http://aclweb.org/anthology/N/N15/N15-1098.pdf`. (Cited on page 32).

Dekang Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, pages 768–774, Montreal, Quebec, Canada, August 1998. Association for Computational Linguistics. doi: 10.3115/980691.980696. URL `http://www.aclweb.org/anthology/P98-2127`. (Cited on pages 16 and 34).

Dekang Lin, Shaojun Zhao, Lijuan Qin, and Ming Zhou. Identifying synonyms among distributionally similar words. In *IJCAI*, volume 3, pages 1492–1493, 2003. (Cited on pages 8, 27, 30, 39, 57, 58, 59, 60 and 61).

Ganna Volodymyrivna Lobanova. *The anatomy of antonymy: a corpus-driven approach.* PhD thesis, 2012. Relation: http://www.rug.nl/ Rights: University of Groningen. (Cited on pages 10, 29, 30 and 40).

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013. URL `http://arxiv.org/abs/1301.3781`. (Cited on pages 33 and 35).

George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995. ISSN 0001-0782. doi: 10.1145/219717.219748. URL `http://doi.acm.org/10.1145/219717.219748`. (Cited on pages 8 and 29).

George A. Miller and Walter G. Charles. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28, 1991. doi: 10.1080/01690969108406936. URL `http://dx.doi.org/10.1080/01690969108406936`. (Cited on page 16).

Saif Mohammad, Bonnie Dorr, and Graeme Hirst. Computing word-pair antonymy. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 982–991, Honolulu, Hawaii, 2008. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1613715.1613843`. (Cited on page 29).

Saif Mohammad, Bonnie J. Dorr, Graeme Hirst, and Peter D. Turney. Computing lexical contrast. *CoRR*, abs/1308.6300, 2013. URL `http://arxiv.org/abs/1308.6300`. (Cited on pages 8, 27, 33, 34, 36, 59 and 66).

Lynne Murphy. *Semantic relations and the lexicon: Antonymy, synonymy and other paradigms.* Cambridge University Press, 2003. (Cited on page 10).

Masataka Ono, Makoto Miwa, and Yutaka Sasaki. Word embedding-based antonym detection using thesauri and distributional information. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 984–989, Denver, Colorado, May–June 2015. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/N15-1100`. (Cited on pages 29 and 59).

J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, 1993. (Cited on page 23).

Jason D. Rennie, Lawrence Shih, Jaime Teevan, and David R. Karger. Tackling the poor assumptions of naive bayes text classifiers. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 616–623, 2003. URL `http://www.aaai.org/Library/ICML/2003/icml03-081.php`. (Cited on page 21).

Michael Roth and Sabine Schulte im Walde. Combining word patterns and discourse markers for paradigmatic relation classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL), Volume 2*, pages 524–530, Baltimore, Maryland, USA, 22–27 June 2014. URL `http://aclweb.org/anthology//P/P14/P14-2086.pdf`. (Cited on page 8).

Enrico Santus, Qin Lu, Alessandro Lenci, and Churen Huang. Unsupervised antonym-synonym discrimination in vector space. In *Proceedings of the First Italian Conference on Computational Linguistics CLiC-it 2014 and of the Fourth International Workshop EVALITA 2014*, 2014. (Cited on pages 27, 28, 35 and 61).

Silke Scheible, Sabine Schulte im Walde, and Sylvia Springorum. Uncovering distributional differences between synonyms and antonyms in a word space model. In *Proceedings of the 6th International Joint Conference on Natural Language Processing*, pages 489–497, 2013. (Cited on pages 27 and 28).

Sabine Schulte im Walde and Maximilian Köper. Pattern-based distinction of paradigmatic relations for German nouns, verbs, adjectives. In Iryna Gurevych, Chris Biemann, and Torsten Zesch, editors, *Language Processing and Knowledge in the Web*, volume 8105 of *Lecture Notes in Computer Science*, pages 184–198. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-40721-5. doi: 10.1007/978-3-642-40722-2_19. URL `http://dx.doi.org/10.1007/978-3-642-40722-2_19`. (Cited on pages 27, 28 and 30).

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1297–1304. MIT Press, 2005. URL `http://papers.nips.cc/paper/2659-learning-syntactic-patterns-for-automatic-hypernym-discovery.pdf`. (Cited on pages 8, 30 and 40).

György Szarvas, Chris Biemann, and Iryna Gurevych. Supervised all-words lexical substitution using delexicalized features. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1131–1141, Atlanta, Georgia, USA, June 2013. (Cited on page 39).

Peter D. Turney. A uniform approach to analogies, synonyms, antonyms, and associations. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 905–912, 2008. (Cited on pages 27, 28, 30, 57, 58 and 61).

Ellen Vorhees and David Graff. Aquaint-2 information-retrieval text research collection ldc2008t25. *Web download. Philadelphia: Linguistic Data Consortium*, 2008. (Cited on page 32).

Wen-tau Yih, Geoffrey Zweig, and John C. Platt. Polarity inducing latent semantic analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1212–1222, Jeju Island, Korea, 2012. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=2390948.2391085`. (Cited on page 29).

George K. Zipf. Relative frequency as a determinant of phonetic change. *Reprinted from the Harvard Studies in Classical Philology*, 1929. URL `www.jstor.org/stable/310585`. (Cited on page 41).