Technical Report

UCAM-CL-TR-840
ISSN 1476-2986

Number 840

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

Minimally supervised dependency-based methods for natural language processing

Marek Rei

September 2013

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

This technical report is based on a dissertation submitted December 2012 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Churchill College.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

Abstract

This work investigates minimally-supervised methods for solving NLP tasks, without requiring explicit annotation or training data. Our motivation is to create systems that require substantially reduced effort from domain and/or NLP experts, compared to annotating a corresponding dataset, and also offer easier domain adaptation and better generalisation properties.

We apply these principles to four separate language processing tasks and analyse their performance compared to supervised alternatives. First, we investigate the task of detecting the scope of speculative language, and develop a system that applies manually-defined rules over dependency graphs. Next, we experiment with distributional similarity measures for detecting and generating hyponyms, and describe a new measure that achieves the highest performance on hyponym generation. We also extend the distributional hypothesis to larger structures and propose the task of detecting entailment relations between dependency graph fragments of various types and sizes. Our system achieves relatively high accuracy by combining distributional and lexical similarity scores. Finally, we describe a self-learning framework for improving the accuracy of an unlexicalised parser, by calculating relation probabilities using its own dependency output. The method requires only a large in-domain text corpus and can therefore be easily applied to different domains and genres.

While fully supervised approaches generally achieve the highest results, our experiments found minimally supervised methods to be remarkably competitive. By moving away from explicit supervision, we aim to better understand the underlying patterns in the data, and to create systems that are not tied to any specific domains, tasks or resources.

Acknowledgements

Writing a dissertation is a tremendous task, and it would not have been possible without the help of others. I would first like to thank my supervisor, Professor Ted Briscoe, for invaluable guidance and advice throughout the completion of this doctorate. He was able to give me great freedom for pursuing interesting research directions, while our regular meetings and discussions provided excellent insight and kept me firmly on track.

Being part of the Natural Language and Information Processing research group has been an important influence, and I am grateful to all the people I had the chance to collaborate with. I would like to thank Stephen Clark and Anna Korhonen for giving early feedback on my work in progress, Andreas Vlachos for very interesting discussions, and Diarmuid Ó Séaghdha for reading my thesis and providing valuable suggestions. I am very thankful to my examiners, Ann Copestake and Rob Gaizauskas, who helped me with knowledgeable comments and feedback.

I am also very grateful to the Computer Laboratory, Churchill College and EPSRC for funding my studies, making it possible for me to pursue a degree in Cambridge.

I would especially like to thank Helen Yannakoudakis, who has given me constant advice and encouragement. Her invaluable support, both intellectual and personal, has helped me in so many different ways.

Finally, I am thankful to my family – my mother Sirje, father Karl, and sister Merlyn – who have always been there for me. Completing this work would not have been possible without their love and support.

Contents

Chapter 1

Introduction

Natural Language Processing (NLP) is a general term for a wide range of tasks and methods related to automated understanding of human languages. In recent years, the amount of available diverse textual information has been growing rapidly, and specialised computer systems can offer ways of managing, sorting, filtering and processing this data more efficiently. As a larger goal, research in NLP aims to create systems that can also 'understand' the meaning behind the text, extract relevant knowledge, organise it into easily accessible formats, and even discover latent or previously unknown information using inference. For example, the field of biomedical research can benefit from various text mining and information extraction techniques, as the number of published papers is increasing exponentially every year, yet it is vital to stay up to date with all the latest advancements.

Research in Machine Learning (ML) focuses on the development of algorithms for automatically learning patterns and making predictions based on empirical data, and it offers useful approaches to many NLP problems. Machine learning techniques are commonly divided into three categories:

1. **Supervised** learning methods make use of labelled training data to build models that can generalise to unseen examples. These include many well-known learning algorithms, such as support vector machines (Cortes & Vapnik, 1995; Joachims, 1998), conditional random fields (Lafferty *et al.*, 2001), probabilistic neural networks (Specht, 1990) and random forests (Breiman,

2001).

2. **Semi-supervised** systems normally require smaller amounts of labelled data, and also make use of some unlabelled corpora. This can be achieved by bootstrapping the system with a small set of annotated examples and iteratively finding more from the unlabelled data (Agichtein & Gravano, 2000; Thelen & Riloff, 2002), or by propagating the labels of known examples to unseen instances using some similarity metric (Zhu & Ghahramani, 2002; Chen *et al.*, 2006).

3. **Unsupervised** learning methods aim to find a hidden structure in the provided dataset, without using any explicit labelling information. Due to the restrictions of the problem, this usually reduces to some form of clustering, such as k-means (Lloyd, 1982), hierarchical clustering (Johnson, 1967), or self-organising maps (Kohonen, 1990).

Supervised systems have been shown to deliver state-of-the-art performance on a variety of tasks. They are able to automatically learn complicated patterns from the data, and if a suitable collection of annotated examples is available, supervised techniques can offer very competitive results for most NLP challenges. However, obtaining this annotation can be a very costly and time-consuming process. Manual annotation is often a slow and laborious task, and the datasets need to be large enough to allow for sufficient generalisation by the algorithms, usually thousands or tens of thousands of examples. NLP tasks are often domain-specific, and therefore the annotation should be performed by domain experts, who can sometimes be reluctant to donate their time. Furthermore, in order to minimise bias and error, the data should be cross-annotated by more than one expert. Finally, when the dataset is created, it can be utilised to train a model for a specific task, but it is usually unable to generalise to different tasks, or even different domains of the same problem.

In this work we investigate **minimally-supervised** methods for solving NLP tasks, and we define them as requiring no explicit training data. Since completely unsupervised systems are usually only applicable to tasks that can be formulated as clustering problems, we expand this definition by also allowing:

a) existing NLP tools (e.g., tokenisers, POS taggers, parsers) that are not specific to our task or domain, and/or

b) domain-specific resources, other than annotated training data, which could be produced by a domain-expert in a relatively short time.

While the amount of required supervision depends on the specific task, we aim to develop approaches that are expected to require only a fraction of work from domain experts, compared to manually annotating a dataset for supervised learning. Semi-supervised systems approach similar problems by making use of some unlabelled data, but they still require varying amounts of annotation. These methods have been more thoroughly investigated by other researchers (Zhu, 2005; Chapelle *et al.*, 2006; Vlachos, 2010), and they will not be the focus in this work.

For developing minimally-supervised systems, we make use of automatically generated dependency graphs, allowing us to find the binary semantic relations between individual words. In contrast to parse trees that show the syntactic construction and grammatical rules in a sentence, dependency relations provide a more intuitive representation of the meaning behind sentences. For example, the edge labels in Figure 1.1 indicate that *design* is the core of the sentence (missing head relation), *engineers* are the people doing the designing (non-clausal subject relation), and *engines* are being designed (direct object relation). Dependency graphs are also better suited for representing discontinuous constructions, such as long-distance dependencies, or grammatical relations that are signalled by morphology instead of word order (McDonald & Nivre, 2011). By recursively traversing the directed graph, a system can automatically find how individual words relate to each other, and reconstruct the semantics of the whole sentence.

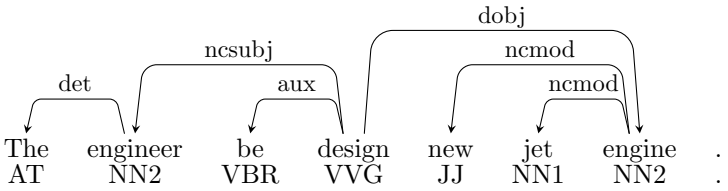

Figure 1.1: Example dependency graph for the sentence: *The engineers are designing new jet engines.*

Dependency parsers are available for many different languages and domains, and they are being utilised to solve various tasks with increasing frequency, making them one of the most commonly used tools in natural language processing. Our work and methods focus on novel uses of dependency graphs, as they provide valuable information about the semantics of the text, even without any available task-specific annotation. We experiment with defining heuristic rules over the graph structure (Chapter 3), use dependency graphs to extract features for a supervised classifier (Chapter 3), collect distributional features from dependency graphs (Chapter 4), extend the distributional hypothesis to dependency graph fragments (Chapter 5), and improve the accuracy of a parser by having it learn from its own dependency output (Chapter 6).

The goal of this work is to create systems that require substantially reduced effort from domain experts, compared to annotating a corresponding dataset. Our research is largely motivated by potential applications in the biomedical domain, where expertly annotated data is difficult to obtain and does not exist for many NLP tasks. However, lack of annotated data also prevents us from performing conclusive evaluation and comparison of all our methods in this domain. Therefore, we conduct experiments on biomedical data whenever possible, but also make use of other domains and datasets, where annotation is more readily available.

Although many problems can be represented as sets of feature vectors in a very straightforward way, we believe it is important not to get overly reliant on using only supervised classifiers. A famous quote by Abraham Maslow says "I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail." (Maslow, 1968). By moving away from explicit supervision, we aim to better understand the underlying patterns in the data, and to create systems that are not tied to a specific domain or dataset. In addition, the output of such methods is likely to have good discriminative properties, and can therefore be used to directly benefit fully supervised systems.

1.1 Research goals

In this work, we investigate four different NLP tasks and approach them using minimally supervised dependency-based methods. Each of these tasks contains a unique set of challenges and goals that need to be met, and these are described in detail in their corresponding chapters. However, we also have a set of common goals that we follow throughout this thesis:

1. **Investigate minimally supervised methods and how they can be applied to different tasks in natural language processing.** We aim to determine whether minimal supervision is a viable approach to NLP, and whether it can deliver competitive performance even without training data.

2. **Compare the performance of minimally supervised systems to fully supervised systems.** We contrast systems with different levels of supervision in order to measure the decrease in performance when labelled data is not available, thereby quantifying the actual benefit of manual annotation.

3. **Examine new and innovative applications for dependency graphs**. We experiment with various ways of using dependency graphs, illustrating their usefulness as a general-purpose resource for natural language processing, and examining how they can help solve specific language processing tasks.

4. **Find ways of integrating minimally supervised and fully supervised systems.** We determine whether minimally supervised methods can be successfully used as discriminative features in a supervised classifier, and how this affects performance.

5. **Create accurate practical systems for solving each of the tasks.** We analyse every individual challenge in detail and look for underlying patterns. Our focus is on developing solutions that require no training data, thereby making them more robust, more easily adapted to different domains, and also more suitable to be deployed in practical applications.

1.2 Project overview

We investigate the effect of minimally supervised methods by applying them to four separate language processing tasks. In each of these case studies, our goal is to develop a competitive working system for solving the task without requiring training data. This allows us to study the performance of different systems, in order to find common properties and approaches that are capable of generalising across different problem sets and domains. Whenever possible, results are also presented using a supervised system, allowing us to analyse and compare the two alternative methodologies.

The tasks we investigate are as follows:

1.2.1 Detection of speculative language

Speculative language is an important tool in scientific literature, allowing scientists to guide research beyond the evidence without overstating what follows from their work. Vincze *et al.* (2008) show that 19.44% of all sentences in the full papers of the BioScope corpus contain speculation cues. Detecting the scope of uncertainty in text can be important for tasks such as scientific information extraction or literature curation, as typically only definite information should be extracted and curated.

We develop and compare three different approaches to the scope detection task: the baseline system assigns scopes only based on surface text and requires barely any processing; the heuristic system uses a small number of linguistic rules defined over dependency graphs, but requires no annotated data; the supervised system builds on the previous approaches and employs a classifier trained on human-annotated data. Most of this work has been published in the proceedings of the Conference on Natural Language Learning 2010 (Rei & Briscoe, 2010).

1.2.2 Hyponym detection and generation

Automatic detection and generation of hyponyms has many practical applications in nearly all natural language processing tasks. Information retrieval, information

extraction and question answering can be improved by performing appropriate query expansion. Summarisation systems can increase coherence and reduce repetition by correctly handling hyponymous words in the input text. Entailment and inference systems can improve sentence-level entailment resolution by detecting the presence and direction of word-level hyponymy relations. Hyponyms are also useful for smoothing language models and word co-occurrence probabilities.

We investigate how hyponyms can be detected and generated based on their distributional similarity, using dependency relations as context features and requiring only a large corpus of plain text for collecting co-occurrence statistics. While most common similarity measures are symmetric, recent work on directional distributional similarity measures has been trying to model asymmetric relations as well. We present a comprehensive comparison of all prominent similarity measures, create several new datasets to evaluate this task, and also propose a novel measure based on our observations.

1.2.3 Fragment entailment detection

By generalising the hyponym relation to larger text units such as phrases, predicates and full sentences, we can derive the relation of entailment. For example, the phrase *argues against* also entails *does not support*. While there exist numerous sentence-level entailment detection systems, a broader approach would allow for entailment discovery among a wider range of fragment types for which no specialised systems exist. In addition, entailment detection between fragments is a vital step towards being able to generate entailing sentences for a given input text.

We present a unified framework that can be used to detect entailment relations between fragments of various types and sizes. The system is designed to work with anything that can be represented as a dependency graph, including single words, constituents of various sizes, text adjuncts, predicates, relations and full sentences. The approach is unsupervised and uses intrinsic similarity, multi-level extrinsic similarity and the detection of negation and hedged language to assign a confidence score to entailment relations between two fragments. Most of this work has been published in the Workshop on Biomedical Natural Language Processing

2011 (Rei & Briscoe, 2011).

1.2.4 Parser lexicalisation

Lexical features are important when choosing the correct derivation in ambiguous contexts, but they have been shown not to transfer well between different domains and genres (Sekine, 1997; Gildea, 2001). At the same time, unlexicalised parsers are surprisingly competitive with their lexicalised counterparts (Klein & Manning, 2004; Petrov *et al.*, 2006). Instead of trying to adapt a lexicalised parser to a new domain, we explore how lexical features can be integrated with an unlexicalised parser.

We use the unlexicalised RASP parser to process a large corpus of in-domain text. Next, we create an augmented version of the dependency graphs by applying a series of linguistically-motivated modifications. Finally, we use scores derived from the corpus statistics to rerank the alternative parses created by RASP, thereby improving the parsing accuracy. The system also makes use of automatically generated hyponyms to smooth the probabilities and improve the modelling of rare relations. This framework allows the unlexicalised parser to learn from its own output and significantly increase accuracy without requiring any additional annotated data. Most of this work has been published in the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013) (Rei & Briscoe, 2013).

Chapter 2

Resources

In this chapter we provide a description of various tools and datasets which we utilise or reference in our research.

2.1 Tools

2.1.1 The RASP System

The Robust Accurate Statistical Parsing (RASP) System[1] (Briscoe *et al.*, 2006; Watson *et al.*, 2005; Watson, 2006, 2007; Andersen *et al.*, 2012) contains modules for a series of language processing tasks, such as finding sentence boundaries, analysing words to identify their root and any suffixes, assigning part-of-speech labels, and analysing the grammatical relations between words within sentences. We employ this toolkit in all of our experiments, as it utilises an unlexicalised statistical parser, thus reducing the required manual annotation and making it less domain-dependent.

The structural parse ranking model is trained (in a semi-supervised fashion) on partially-bracketed sentences from the SUSANNE corpus, itself a subset of the Brown corpus. The RASP system also utilises a part-of-speech (POS) tagger which provides tag sequences to the parser, but this too has been trained on non-WSJ (Wall Street Journal) text drawn from balanced corpora. Although the tagger does make use of lexical statistics to assign POS tags, no further lexical

[1] ilexir.co.uk/applications/rasp/download

information is exploited in the default parse selection model, which we deploy in our experiments. Finally, the parser outputs ranked dependency graphs using the same grammatical relation (GR) scheme (Briscoe & Carroll, 2006) used to annotate the DepBank/GR subset of WSJ section 23 from the Penn Treebank.

The dependency graphs are directed, connected (except in the event of parse failures), but not projective or acyclic. For example, with 'control' relations, such as those governing the implicit subject of an infinitive verbal complement, a dependent of the main verb may also be a dependent of the embedded infinitive, violating the single-head constraint of dependency tree representations. In non-subject relative clauses without an overt relative pronoun, the modified head governs the embedded verb of the relative clause but, in turn, depends on this verb as its object, introducing a cycle in the dependency graph.

The RASP parser is a generalised LR (GLR) parser which builds a non-deterministic LALR(1) parse table from the grammar (Tomita, 1987). A context-free backbone is automatically derived from a unification grammar. The residue of features not incorporated into the backbone are unified on each reduce action and if unification fails the associated derivation paths also fail. The parser creates a packed parse forest represented as a graph-structured stack.

Inui *et al.* (1997) describe the probabilistic GLR (PGLR) model utilised in the RASP system where a transition is represented by the probability of moving from one stack state, σ_{i-1}, (an instance of the graph structured stack) to another, σ_i. They estimate this probability using the stack-top state s_{i-1}, next input symbol l_i and next action a_i. This probability is conditioned on the type of state s_{i-1}. S_s and S_r are mutually exclusive sets of states which represent those states reached after shift or reduce actions, respectively.

Watson *et al.* (2007) utilised about 4,000 unlabelled partially-bracketed training sentences from the SUSANNE corpus to train the PGLR model in a semi-supervised two-step process. An initial model was trained from the subset of about 1,000 bracketed sentences for which there was a single derivation consistent with the bracketing. This initial model was used to rank derivations consistent with the bracketing from the remaining ambiguous portion of the training data. Then the final model was trained using rank-weighted counts from the entire training corpus.

It is difficult to accurately quantify the amount of work required to construct the grammar rules and partial bracketing necessary for training the models in RASP. However, it is definitely less compared to the construction of the Penn Treebank, which is utilised for training most other supervised parsing models. In our experiments we treat RASP as an external tool and employ it on text from different domains, such as biomedical papers and WSJ articles. It is clear that RASP is in no way tuned to perform well in the biomedical domain, as it was developed using the Brown corpus. This might be more debatable for the WSJ experiments, as the Brown corpus covers a wide selection of texts, including news articles from the financial domain. However, financial texts are a relatively small portion of the dataset (only 4 out of 500 samples), and Gildea (2001) also treats the Brown corpus as out of domain with respect to the WSJ.

2.1.2 C&C Tools

The C&C Tools[1] (Clark & Curran, 2004; Curran *et al.*, 2007) consist of the C&C Combinatory Categorial Grammar (CCG) parser, the computational semantics tool Boxer, and the C&C taggers. The grammar used for the parser, consisting of 425 lexical categories expressing subcategorisation information and a small number of combinatory rules, was extracted from CCGBank (Hockenmaier, 2003), a CCG version of the Penn Treebank. A supertagger is first used to assign lexical categories to each word in the sentence, and these are then combined by the parser using the combinatory rules and the CKY algorithm. A log-linear model is used to score alternative parses, with features defined over local parts of the derivation, including word-word dependencies. The parser outputs predicate-argument dependencies defined in terms of CCG lexical categories. In Chapter 6 we compare the performance of the unlexicalised RASP parser, the fully-lexicalised C&C parser, and our self-learning lexicalisation method on the DepBank/GR dataset.

2.1.3 Support Vector Machines

Support Vector Machines (SVM) (Vapnik, 1982; Cortes & Vapnik, 1995) are a type of supervised large-margin discriminative classifier. A binary SVM takes

[1] svn.ask.it.usyd.edu.au/trac/candc

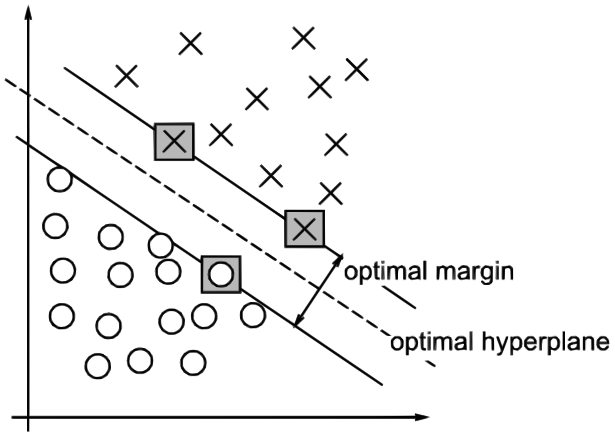

Figure 2.1: Illustration of Support Vector Machines. The circles and crosses represent different classes, the grey squares represent support vectors on the edge of the separation margin. Image by Cortes & Vapnik (1995).

as input a labelled dataset with two classes and learns a model for predicting the class of unseen examples. More specifically, the algorithm finds the hyperplane between two classes which, in addition to separating them, also has the largest distance from either class. The decision function is based on the subset of training examples that lie closest to the separation plane, referred to as support vectors. This strategy allows the model to achieve better performance on unseen data, as the maximum margin hyperplane is expected to correspond to a natural separation between the classes.

While the original algorithm assumes that a clean separation of the two classes is possible, Cortes & Vapnik (1995) introduced soft-margin SVMs that try to find the best possible hyperplane but also allow for misclassified examples. SVMs can also make use of the *kernel trick* which allows solving non-linear problems through linear optimisation techniques.

We utilise SVM^{Light} (Joachims, 1999),[1] a well-known efficient implementation of SVMs, in Chapter 4 to perform supervised learning experiments for hyponym detection and generation.

[1] www.cs.cornell.edu/People/tj/svm_light/

2.1.4 Conditional Random Fields

Conditional Random Fields (CRF) (Lafferty *et al.*, 2001) are a class of probabilistic models for segmenting and labelling sequence data. A CRF can be thought of as an undirected graphical model, globally conditioned on the sequence of observations. This makes them well-suited for classification tasks where labels are highly dependent on sequence and context, such as POS tagging.

Assuming that the graphical model has a simple first-order chain structure, the overall probability of the label sequence can be calculated using a set of transition and state feature functions (Wallach, 2004). We make use of CRF++,[1] an implementation of CRFs, to identify speculation cues and scopes in Chapter 3.

2.2 Developed libraries

In order to perform all the experiments necessary for this research, we created two new software libraries which we are releasing as open-source projects:

2.2.1 SemGraph

SemGraph is a Java library for reading, writing and visualising dependency graphs in different formats.

Numerous alternative dependency parsers have been developed, and most of them specify a unique format for their output. Applications that need to iterate through large collections of parsed data often employ an integrated approach, and the modules for processing parser output are mixed together with system logic. This results in complex systems that are highly parser-dependent, and modifying them to work with different parser formats requires expensive large-scale changes.

With SemGraph we have aimed to develop a library that allows for easy separation of system logic from any specific parser format. In addition, we wish to provide well-tested collaboratively developed methods for reading various formats, instead of every developer re-implementing a personal version of this common

[1] code.google.com/p/crfpp/

```
1 GraphReader reader = new RaspXmlGraphReader(
      "examples/raspxml/file1.xml",
      RaspXmlGraphReader.NODES_TOKENS, false, false);
2 while(reader.hasNext()){
3           Graph graph = reader.next();
4           for(Node node : graph.getNodes())
5                   System.out.println(node.getLemma() +
                        node.getPos());
6 }
```

Figure 2.2: Example Java code for iterating through a corpus in RASP XML format and printing out information about each node.

code. The library can be easily included in any Java project, and the system will be able to use generic functions for accessing dependency graphs, without explicit knowledge of the underlying input format.

For example, a common scenario for an NLP system is to iterate through a large corpus of parsed text in order to collect distributional features or look for specific patterns in dependency graphs. Using SemGraph, the same system can utilise corpora parsed with both RASP and C&C parsers without any modifications, by only specifying the appropriate input location. This enables researchers and developers to easily switch between alternative parsers and corpora, in order to use the most appropriate parser for a specific domain or task. In addition, it reduces the amount of code required for each system, and allows for easy comparison of different parsers.

Figure 2.2 contains an example of working code for reading and processing a corpus of dependency graphs. In line 1 the GraphReader object is initialised with the input path; the program then loops through all the dependency graphs (line 2) and every node in every graph (line 4), printing out the lemma and POS tag (line 5). Currently, the following input formats are supported:

- rasp – The default output format of the RASP parser.
- raspxml – The xml output format of the RASP parser.
- cnc - The default output format of the C&C parser.

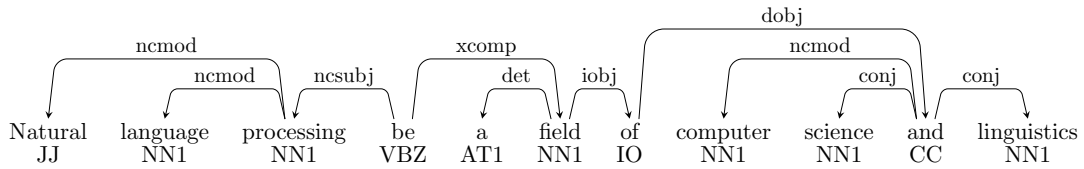

Figure 2.3: The LaTeX version of a dependency graph from RASP.

- parseval - The format used by the Depbank/GR dataset and one of the output formats of the RASP parser.
- tsv - A simple tab-separated format for representing graphs and sentences.

The input can be a single file or a whole directory, containing plain text or gzipped files. We plan to extend the library with support for several other well-known dependency parsers, and we also welcome contributions from other developers.

While the dependency graphs are read into a common data structure, they are not modified in any way; combining output from different parsers can lead to unexpected results, since various sources may use different dependency labels or even have different attachment preferences. Therefore, the library is best used for switching between alternative input types, or combining them only in cases where the differences are less important (e.g., when collecting distributional features).

The SemGraph library also provides methods for writing dependency graphs in different formats. Performing a full conversion from one parser output specification to another would be a useful feature, but this is currently not possible – most parsers include unique fields in their output that are not available from other parsers, and we do not attempt to capture or replicate this highly specific information. However, we are able to offer two output formats:

- tsv - A simple tab-separated format.
- tikzdependency - A LaTeX representation of the graph.

The *tsv* format is supported for both reading and writing, allowing users to modify, save and reload any dependency graphs. The tikzdependency format contains LaTeX code which can be compiled into a pdf image of a dependency

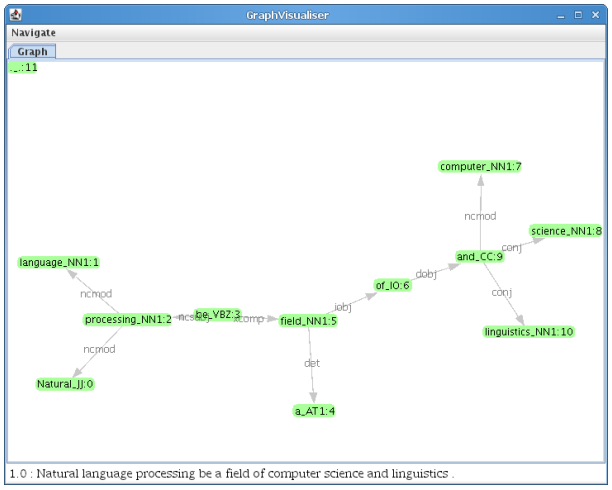

Figure 2.4: The GraphVisualiser tool displaying a dependency graph from RASP.

graph, as shown in Figure 2.3. This allows users to automatically convert parser output into images suitable for research papers or technical reports, without having to handle graphics or layout.

The library also includes a tool for visualising dependency graphs, in order to provide researchers with an intuitive method for navigating parsed corpora and finding patterns in graphs. The GraphVisualiser tool, shown in Figure 2.4, creates a dynamic view of dependency graphs by displaying labelled nodes and edges using a force-directed algorithm. The original sequence of lemmas is provided under the graph, and the user can iterate forward and backward through a large corpus. It also includes support for alternative parses – while keys X and Z step through sentences, keys A and S switch between different parses of the same sentence.

The SemGraph library is an open-source project and available for download.[1]

2.2.2 SemSim

SemSim is a Java library for building distributional models from large corpora of parsed text, and for using these models in various tasks. It is built using the SemGraph library and can therefore use any input format described in Section

[1] github.com/marekrei/semgraph

2.2.1. Current functionality of the library includes:

- Building a three-dimensional co-occurrence tensor from all dependency relations in the corpora, which allows for fast access to various statistics regarding the dataset. For example, we can find how many times does the relation triple *(play, dobj, guitar)* appear in the corpus, or what is the probability of *guitar* being a dependent in a relation.

- Constructing distributional context vectors for each word. Application of various weighting schemes, such as mutual information, is also supported.

- Applying automatic modifications to dependency graphs. The pipeline includes modules for adding graph editors to perform tasks such as text normalisation or adding new edges to the graph using linguistic rules. These modifications can decrease noise and increase the number of available binary relations, leading to more accurate systems. Several graph editors have already been implemented, described in Chapter 6, and new ones can be easily added.

- Calculating distributional similarity between words using a wide range of similarity measures. We have implemented 30 different well-established measures from previous research, and we provide the code for performing experiments with hyponym detection and generation, as described in Chapter 4. The implementation uses caching and a thread pool to evenly distribute tasks between available processors while using the same distributional model, greatly reducing the running time for experiments.

- Reranking dependency parses based on corpus statistics, as described in Chapter 6. We use a distributional model to calculate probabilities for every dependency relation, combine them together into parse scores for reranking, and find distributionally similar words to smooth the calculations. The implementation has also been optimised to make use of multi-core systems for faster processing.

The SemSim library is an open-source project and publically available for download.[1]

2.3 Datasets

2.3.1 British National Corpus

The British National Corpus (BNC) (Burnard, 2007) is a corpus containing 100 million words of written and spoken language from a wide range of sources. It is designed to represent a wide selection of British English from the later part of the 20th century. 90% of the corpus consists of written text, such as extracts from newspapers, specialist periodicals and journals, academic books and popular fiction, published and unpublished letters, school and university essays, and many other kinds of text. The remaining 10% contains orthographic transcriptions of unscripted informal conversations and spoken language collected in different contexts. The corpus encoding also contains automatically added part-of-speech tags and various structural information about the text, such as headings, paragraphs and lists.

Work on the BNC started in 1991, and the first version was completed in 1994, followed by the revised second edition (BNC World) in 2001, and the third edition (BNC XML) in 2007. In our experiments we make use of the third edition of the BNC parsed with RASP4UIMA (Andersen *et al.*, 2008).

2.3.2 BLLIP WSJ Corpus

The Brown Laboratory for Linguistic Information Processing (BLLIP) 1987-89 WSJ Corpus Release 1 (Charniak *et al.*, 2000) contains approximately 30 million words of text extracted from the Wall Street Journal (WSJ) over a three-year period (1987 – 1989). The entire archive has been automatically parsed and part-of-speech (POS) tagged using statistical methods, but the annotation has not been manually checked. This corpus is a superset of the million-word Penn Treebank (PTB) collection of parsed and POS-tagged WSJ texts. However, our

[1] github.com/marekrei/semsim

version of the BLLIP WSJ corpus excludes any texts found in PTB, as they are used to evaluate our parse reranking framework.

2.3.3 Brown Corpus

The Brown University Standard Corpus of Present-Day American English (Brown Corpus) (Francis & Kučera, 1979) contains 1,014,312 words of edited English-language prose. The texts come from a variety of sources, all published in 1961 in the United States and written by native speakers, as far as could be determined. The Corpus is divided into 500 samples of at least 2000 words each, with every sample beginning at the start of a new sentence, but not necessarily of a paragraph or other larger division. The samples are distributed across 15 genres (e.g., *Press Reportage*, *Press Editorial*, and *Science Fiction*), roughly in the same proportions as the amount published in 1961. A later version of the corpus also contains POS tags, which were added automatically by a tagger, but have also been manually checked over the course of several years.

2.3.4 SUSANNE Corpus

The Surface and Underlying Structural Analysis of Naturalistic English (SUSANNE) Corpus (Sampson, 1995) is an annotated sample comprising approximately 130,000 words of written American English text. It was created as part of an effort towards developing a comprehensive annotation scheme for the logical and surface grammar of English. SUSANNE guidelines aim to specify an explicit notation for all aspects of English grammar in sufficient detail, such that independent analysts working on the same text would produce identical annotations. The texts included in the corpus are a subset of 64 sample texts from the Brown Corpus, covering 4 different genres: *Press Reportage*, *Belles Lettres* (biographies and memoirs), *Learned* (scientific and technical writing), *Adventure and Western Fiction*.

2.3.5 Penn Treebank

The Penn Treebank (PTB) (Marcus *et al.*, 1993, 1994; Taylor *et al.*, 2003), also known as UPenn Wall Street Journal (WSJ) Treebank, contains approximately 50,000 sentences (1 million words) manually annotated with part-of-speech tags and phrase-structure trees. The texts originate from 1989 Wall Street Journal Articles (Dow Jones Newswire stories). The initial release was created over a period of three years and contains annotation of only skeletal trees with syntactic structure. The annotation scheme was later updated (Treebank-2) to allow for extraction of simple predicate-argument structure and include POS-tags. In the later stages, the Treebank project also produced a tagged and parsed version of the Switchboard corpus of transcribed telephone conversations, including annotation for common disfluencies in speech (Treebank-3).

2.3.6 PARC 700 Dependency Bank

The PARC 700 Dependency Bank (DepBank) (King *et al.*, 2003) contains 700 sentences randomly selected from section 23 of the WSJ Penn Treebank. They were parsed with a broad coverage LFG grammar of English using the XLE system (Maxwell III & Kaplan, 1993), automatically converted to DepBank format, and then manually corrected and extended by human validators. The final annotation contains syntactic features and bilexical head-dependent relations derived from the F-structure representation of LFG.

2.3.7 DepBank/GR

DepBank/GR (Briscoe & Carroll, 2006) is a reannotated version of the PARC 700 Dependency Bank (DepBank), with the labelling matching the GR output format of the RASP parser (Briscoe *et al.*, 2006). The GR scheme includes one matching feature from DepBank (passive); several DepBank relations have been split into separate relations for the GR scheme (e.g., adjunct); some relations from DepBank are merged together (e.g., oblique and iobj); the GR scheme also includes some additional featural and subtype information. DepBank/GR is the main evaluation dataset for RASP and is distributed together with the parser.

The corpus is split into 560 sentences for testing and 140 sentences of held-out/development data.

2.3.8 WordNet

WordNet (Miller, 1995) is a large lexical database of the English language. Nouns, verbs and adjectives are grouped into 117,000 cognitive synonym sets (synsets), and short definitions for each of the synsets are provided. The database also contains annotation for a range of semantic relations between these synsets, for example hyponymy, meronymy, antonymy, and entailment. It is designed for use both as a dictionary/thesaurus, and to be accessed automatically by artificial intelligence or text analysis systems. WordNet has been successfully used as a resource for many NLP tasks, such as topic classification, summarisation, information retrieval, information extraction, machine translation and word sense disambiguation. It contains a sufficiently wide range of commonly used words, but does not include vocabulary for more specific domains, such as the life sciences. We make use of the annotation in WordNet 3.0 to evaluate methods for hyponym detection and generation in Chapter 4.

2.3.9 MEDLINE

MEDLINE (Medical Literature Analysis and Retrieval System Online)[1] is a bibliographic database of journal articles in life sciences with a focus on biomedicine. It contains over 21 million citations from approximately 5,600 worldwide journals, covering the period from 1950 to the present. Among the citations added between 2005 and 2009, roughly 91% are published in English, and 83% have English abstracts written by authors of the articles. The records in MEDLINE are indexed with U.S. National Library of Medicine's (NLM) Medical Subject Headings (MeSH).

[1] www.nlm.nih.gov/pubs/factsheets/medline.html

2.3.10 PubMed

PubMed[1] is a citation database containing over 22 million citations for biomedical and health literature, also covering fields of medicine, behavioural sciences, chemical sciences, preclinical sciences and bioengineering. It contains the records from MEDLINE, and additional articles from selected life science journals. As of September 2012, over 12 million of these articles are listed with their abstracts, and approximately 500,000 new records are added to the database every year. PubMed is also the main method for accessing the records in MEDLINE.

2.3.11 PubMed Central

PubMed Central (PMC)[2] is a free archive of biomedical and life sciences journal literature. As of September 2012, it contains over 2.5 million items, including articles, editorials and letters. Where available, the records in PubMed are cross-linked to their corresponding entries in PubMed Central. The PMC Open Access Subset is a relatively small part of the total collection of articles in PMC, currently containing 497,820 records made available under a Creative Commons or similar licence. The full text of these PubMed Central articles is freely available for bulk downloading and processing, making it a valuable resource for various text mining applications.

2.3.12 BioMed Central

BioMed Central (BMC)[3] is an open-access publisher that publishes a range of journals across different biomedical fields, from basic life sciences to clinical medicine. As of September 2012, BMC has published 133,566 articles of peer-reviewed research, all of which are covered by an open access licence agreement, allowing free distribution and re-use of the full-text articles. All the articles are available for download in structured XML format, and also included in the PubMed Central open access dataset. In our experiments we use an earlier version of the corpus from 2010, containing 71,821 full papers.

[1] www.ncbi.nlm.nih.gov/pubmed
[2] www.ncbi.nlm.nih.gov/pmc/
[3] www.biomedcentral.com

2.3.13 GENIA

The GENIA corpus (Kim *et al.*, 2003) is a collection of 1,999 MEDLINE abstracts annotated with various levels of linguistic and semantic information. The corpus was created to support the development and evaluation of text mining and information extraction systems in the domain of molecular biology. The articles in GENIA were selected using a PubMed query for the three MeSH terms *'human'*, *'blood cells'*, and *'transcription factors'*. Based on the categories of annotation, GENIA is divided into the following subcorpora: part-of-speech annotation, constituency (phrase structure) syntactic annotation, term annotation, event annotation, relation annotation, and coreference annotation.

2.3.14 GENIA-GR

The GENIA-GR dataset (Tateisi *et al.*, 2008) is a corpus created for parser evaluation in the biomedical domain. It is a subset of the GENIA corpus, containing 50 abstracts (492 sentences). The abstracts were chosen to satisfy the following conditions:

1. The abstract is indexed with the MeSH term *'NF-Kappa B'*.
2. The full text of the article is freely available from PubMed Central.
3. The abstract does not belong to the set used for training parsers by Hara *et al.* (2007).

The dataset is annotated using the grammatical relations scheme proposed by Carroll *et al.* (1998) and revised by Briscoe *et al.* (2006), also matching the DepBank/GR dataset described in Section 2.3.7. Sentences were first parsed by the RASP parser and then manually corrected.

2.3.15 BioScope

The BioScope corpus (Vincze *et al.*, 2008) consists of biological and medical texts annotated for negation and speculation, including their linguistic scope. It is designed to allow for comparison of different systems on the tasks of negation/hedge

detection and scope resolution. The corpus is divided into three sections based on the text sources:

- 1,273 abstracts from the GENIA corpus, containing 11,871 sentences.
- 9 full papers, containing 2,670 sentences. 5 of these papers were previously annotated by Medlock & Briscoe (2007), the remaining 4 were selected from the BioMed Central repository.
- 1,954 clinical free texts from the radiology report corpus, used for the CMC clinical coding challenge (Pestian *et al.*, 2007), containing 6,383 sentences.

The CoNLL-2010 shared task on identifying hedges and their scope (Farkas *et al.*, 2010) also incorporated the BioScope corpus, and 15 additional full papers were annotated for evaluation.

Chapter 3

Detection of speculative language

3.1 Introduction

Speculative or "hedged" language is a way of weakening the strength of a statement. It is usually signalled by a word or phrase, called a hedge cue, which weakens some clauses or propositions. These weakened portions of a sentence form the scope of the hedge cues. In the following examples, cues are marked with angle brackets and scopes with round brackets:[1]

(1) These findings (<might> be chronic) and (<may> represent reactive airways disease).

(2) (This subdomain, corresponding to the first exon in the genes split by two introns, <appears> to be missing in the sea urchin).

Hedging is an important tool in scientific language allowing scientists to guide research beyond the evidence without overstating what follows from their work. Vincze *et al.* (2008) show that 19.44% of all sentences in the full papers of the BioScope corpus contain hedge cues. Detecting these cues is potentially valuable for tasks such as scientific information extraction or literature curation, as typically only definite information should be extracted and curated. Most work so far

[1] The difference in annotated scopes for these two examples is somewhat controversial, but it follows the established guidelines of the CoNLL 2010 Shared Task; more information is available in Section 3.7.1

has been done on classifying entire sentences as hedged or not, but this risks losing valuable information in (semi-)automated systems. More recent approaches attempt to find the specific parts of a sentence that are hedged.

The detection of hedge scopes is usually done in two stages – the hedge cues are detected first, and a separate system is then utilised to find the scope of each cue. Both of these tasks can be formulated as a supervised learning problem (e.g., token classification), or by manually constructing linguistically-motivated rules that correctly resolve the scope. In this chapter we describe our implementations of both approaches, and compare the performance of a fully-supervised system to a minimally supervised framework utilising a small number of heuristic rules. We also developed a combination of both methods for the CoNLL 2010 Shared Task: Learning to Detect Hedges and their Scope in Natural Language Text (Farkas *et al.*, 2010). The system is designed to find hedge cues and scopes in biomedical research papers and works in three stages:

1. Detecting the cues using a token-level supervised classifier.
2. Finding the scopes by combining manual rules and a second supervised token-level classifier.
3. Applying postprocessing rules to convert the token-level annotation into predictions about scope.

Our final system achieves 55.9% F-measure on the task of detecting both cues and scopes in the biomedical papers, ranking second in the shared task results. The experiments in this chapter make use of the dataset and evaluation metrics from the shared task, and the results are therefore comparable to all the other submissions.

3.2 Research goals

Our work on detection of speculative language has the following research goals:

- Investigate the relationship between the scopes of speculation cues and the paths in dependency graphs.

- Create a set of heuristic rules for resolving speculation scopes.
- Develop an effective minimally supervised system for detecting speculation cues and scopes.
- Compare the performance of minimally supervised systems to fully-supervised systems on the tasks of detecting speculation cues and scopes.
- Experiment with combining heuristic methods and machine learning for speculation scope detection.

3.3 Background

Hedge cues typically fall into one of the following categories (Farkas *et al.*, 2010):

- auxiliaries: *may, might, can, would, should, could*, etc.
- verbs of hedging or verbs with speculative content: *suggest, question, presume, suspect, indicate, suppose, seem, appear, favour*, etc.
- adjectives or adverbs: *probable, likely, possible, unsure*, etc.
- conjunctions: *or, and/or, either ... or*, etc.

However, some complex phrases can also function as hedge cues – for example, *cannot exclude the possibility, raises the question of.*

The first linguistically and computationally motivated study of hedging in biomedical texts was done by Light *et al.* (2004). They presented an analysis of the problem based on Medline abstracts and constructed an initial experiment for automated classification of speculative text.

Following their work, most research has focused on classifying sentences as hedged or not, rather than finding the scope of the speculation. Medlock & Briscoe (2007) proposed a weakly supervised machine learning approach to the hedge classification problem. Their system utilised a supervised classifier with single words as features and a small amount of seed data for bootstrapping the system. This work was extended by experimenting with a wider range of features

such as part-of-speech tags, lemmas, bigrams, trigrams, and external data sources (Medlock, 2008; Szarvas, 2008). Kilicoglu & Bergler (2008) applied a combination of lexical and syntactic methods, improving on previous results and showing that quantifying the strength of a hedge can be beneficial for classification of speculative sentences. Some of the most recent approaches include using meta-classifiers (Tang *et al.*, 2010), Bayesian logistic regression (Vlachos & Craven, 2010), and high-quality lexical feature selection (Georgescul, 2010).

With the creation of the BioScope corpus (Vincze *et al.*, 2008), an annotated dataset of biomedical papers, abstracts and clinical texts, research focus has begun to shift more towards detecting the individual cues and scopes in sentences. The corpus, described in more detail in Section 2.3.15, was built by extending the dataset and annotation scheme used by Medlock & Briscoe (2007), and was also used as training data for the CoNLL-10 Shared Task.

Morante & Daelemans (2009) were one of the first to approach the problem of identifying cues and scopes in the BioScope corpus using supervised machine learning. They trained a selection of classifiers to tag each word and combined the results with a final classifier, identifying 65.6% of the scopes in abstracts and 35.9% of the scopes in papers. Özgür & Radev (2009) developed a system that detects hedge cues using a supervised token-labelling approach and then resolves the scope by applying manually-constructed rules defined over the parse tree.

One of the CoNLL-10 shared tasks involved the detection of individual hedge cues and scopes in each sentence, and it led to the development of a wide variety of different systems. As one of the best-performing submissions, Morante *et al.* (2010) utilised a range of lexical, context and dependency features together with a memory-based classifier to label each token, and performed post-processing steps to construct speculation scopes. Their classification algorithm is based on the k-nearest neighbour rule and uses a weighted class vote to find the label. In contrast, Velldal *et al.* (2010) deployed a maximum entropy classifier for cue detection and manual rules over dependency representations for scope resolution. Their system was later improved by incorporating additional syntactic features (Øvrelid *et al.*, 2010).

In the most recent work, Apostolova *et al.* (2011) presented a system that automatically learns syntactic patterns over a parse tree for resolving the scope

of both speculation and negation. Read *et al.* (2011) reformulated the problem of scope detection as that of selecting the correct subtree from a parse tree, given a list of candidates; they then rank these candidates using a linear SVM-based scoring function. Velldal *et al.* (2012) describe a method for automatically learning a discriminative ranking function over nodes in constituent trees, and currently achieve the highest reported results on the CoNLL-10 Shared Task dataset.

3.4 Rule-based methods

3.4.1 Speculation cues

The detection of hedge cues can be done by creating a fixed list of word sequences and matching them to the sentences, labelling every occurrence as a cue. This approach is quite straightforward as the number of unique cues is relatively small – our training set contains 3,376 cues but only 142 unique examples.[1] However, a word that acts as a cue is not necessarily a cue in every context. For example, the word *or* occurs 1,215 times and only 146 of them are annotated as hedge cues. Selecting all such words as cues will create a system with high recall but very low precision. Therefore, a more reliable approach would need to select only instances that have a high probability of being cue words.

We first extracted all word sequences that were tagged as cues in the training data. Next, we removed any words that occurred more often as a regular word than a cue. Finally, the list was narrowed down further by removing any cues that occurred only once in the data. The final set of 42 cue words is shown in Table 3.1. While our selection was done using annotated data, a similar list could be manually constructed by a language expert, for example by observing system output and iteratively improving the selection. Omitting some of the less frequent cues would not have substantial impact on the performance – while this set covers 79.98% of all occurrences of speculation cues in the training data, using only half of them would still include 76.39% the cues.

This rule-based system compares each of the word sequences to the input

[1] See Appendix A for a complete list of speculation cues in the training data

appear, assume, believe, cannot be exclude, hope, hypothesise, hypothesize, hypothetically, if not, imply, indicate that, indication, likely, may, might, not clear, not know, perhaps, possible, possibly, potentially, presumably, probable, probably, propose, putative, raise the possibility, remain to be elucidate, seem, should, speculate, suggest, suggestion, suggestive, suppose, suspect, think, unclear, unlikely, whether, whether or not, would

Table 3.1: List of lemmatised hedge cues selected for string matching.

sentences and labels every match as a hedge cue. The method is extremely fast and scalable, as it does not require any POS-tagging, parsing or classification. Experiments in Section 3.7.2 show that it performs surprisingly well compared to much more complex systems.

3.4.2 Speculation scopes

The scope of each cue can also be detected using a set of manual rules. The simplest solution to this task is to tag everything between the cue word and the end of the sentence as belonging to the scope. However, we can do better by observing that the speculation scopes are usually connected subgraphs of dependency graphs. Also, the pattern of the scope often depends on the type of the cue – for example, while the scopes normally start with the cue word, they also include the preceeding subject if the cue verb is passive.

To exploit the information in dependency graphs, we parse the sentences using the RASP parser and develop rules that mark certain words in the directed dependency graph as belonging to the scope, depending on the position and POS tag of the cue node. The terminology we use in the following rules and examples:

- "below" refers to nodes that are in the subgraph of dependencies rooted in the current node.
- "parent" refers to the node that is the head of the current node in the directed, connected dependency graph.
- "before" and "after" refer to word positions in the text, centred on the current node.

- "mark everything below" means mark all nodes in the subgraph as being in the scope. However, the traversal of the graph is terminated when a text adjunct (TA) dependency boundary or a word POS-tagged as a clause separator is found, since they often indicate the end of the scope.
- All the conditions have to be met together. For example "below the parent and after the cue" refers to words that are simultaneously below the parent in the dependency graph and after the cue in the sentence.

The rules for finding the scope of a cue are triggered based on the generalised POS tag of the cue:

- **Auxiliary — VM**
  Mark everything that is below the parent and after the cue.
  If the parent verb is passive, mark everything below its subject (i.e., the dependent of the subj dependency) before the cue.
- **Verb — VV**
  Mark everything that is below the cue and after the cue.
  If the cue is *appear* or *seem*, mark everything below subject before the cue.
  If the cue is passive, mark everything below subject before the cue.
- **Adjective — JJ**
  Find the parent of the cue. If there is no parent, the cue is used instead.
  Mark everything that is below the parent and after the cue.
  If the parent is passive, mark everything below subject before the cue.
  If the cue is *(un)likely* and the next word is *to*, mark everything below subject before the cue.
- **Adverb — RR**
  Mark everything that is below the parent and after the cue.
- **Noun — NN**
  Find the parent of the cue. If there is no parent, the cue is used instead.
  Mark everything that is below the parent and after the cue.
  If the parent is passive, mark everything below subject before the cue.

- **Conjunction — CC**
  Mark everything below the conjunction.
  If the cue is *or* and there is another cue *either* before, combine them together. *Either . . . or . . .* is a frequent exception containing two separate cues that form a single scope.

- **"Whether" as a conjunction — CSW**
  Mark everything that is below the cue and after the cue.

- **Default — anything else**
  Mark everything that is below the parent and after the cue.
  If the parent verb is passive, mark everything below the subject before the cue.

It is worth noting that these rules are designed to match the annotation guidelines of the shared task, and might not therefore match alternative definitions of speculation scope.

A partial dependency graph for Sentence 3 is shown in Figure 3.1 (with positional numbering suppressed for readability).

(3) Lobanov et al. thus developed a sensitive search method to deal with this problem, but they also admitted that it (<would> fail to identify highly unusual tRNAs).

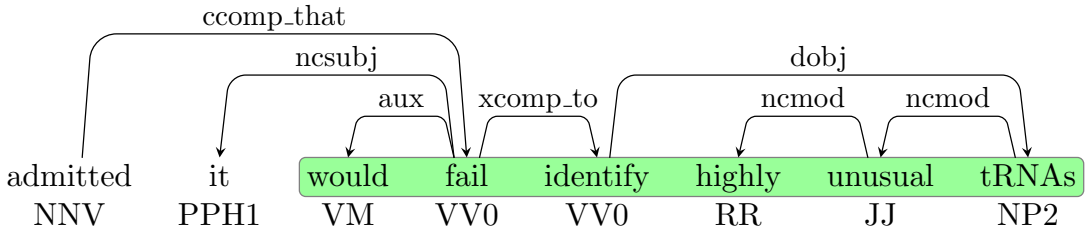

Figure 3.1: Partial dependency graph for sample sentence (3)

Following the rules, *would* is identified as a cue word with the part-of-speech VM, which triggers the first rule in the list. The parent of *would* is *fail* since they are connected with a dependency where *fail* is the head. Everything that is below

fail in the dependency graph and positioned after *would* is marked as being in the scope. Since *fail* is not passive, the subject *it* is left out. The final scope returned by the rule is then *would fail to identify highly unusual tRNAs*.

This method does not require any manually annotated data and only relies on 8 linguistically-motivated rules. Similar approaches could be used for other languages and tasks, such as detecting the scope of negation words. The application of the rules described in this section requires the sentences to be parsed – this means the process is not as fast as the rule-based cue detection step. However, it avoids an additional classification step and is therefore faster than most other scope detection methods.

3.5 Supervised methods

3.5.1 Speculation cues

One of the most common approaches to detecting speculation cues and scopes in natural language texts is to formulate the task as a token-level classification problem. Every word is labelled separately, making use of a wide range of features, such as POS tags, context in the sentence, paths in the dependency graphs, and features extracted from parse trees.

For our supervised system of detecting hedge cues we made use of a Conditional Random Field (CRF) (Lafferty *et al.*, 2001) classifier, implemented as CRF++[1]. The CRF model was chosen because it is known to achieve state-of-the-art performance on related sequence-classification tasks. Each word token is assigned one of the following tags: **F** (first word of a cue), **I** (inside a cue), **L** (last word of a cue), **O** (outside, not a cue), hereafter referred to as the FILO scheme. Minimally two classes (**IO**) are required for this task, while three classes (**FIO**) are also commonly used. We chose to add a fourth class for the last word of the cue, to allow the system to learn more accurate conditions for ending long cue phrases.

The feature types used for classification are defined in terms of the dependency relations output provided by the RASP system. We use binary features that

[1] crfpp.sourceforge.net

indicate whether a word token is a head or a dependent in specific types of relations. This distinguishes between different functions of the same word (when used as a subject, object, modifier, etc.). These features are combined with the POS and lemma of the word to distinguish between uses of different cues and cue types. We also utilise features for the lemma and POS of the 3 words before and after the current word.

The feature types for training the supervised classifier are as follows:

- words (in their inflected forms)
- lemma
- part-of-speech
- broad part-of-speech
- incoming dependencies + POS
- outgoing dependencies + POS
- incoming dependencies + POS + lemma
- outgoing dependencies + POS + lemma
- lemma + POS + POS of next word
- lemma + POS + POS of previous word
- 3 previous lemma + POS combinations
- 3 following lemma + POS combinations.

Outgoing dependencies are relations where the current word is the head, incoming dependencies where it is the dependent.

The predictions from the classifier are compared to the list of known cues extracted from the training data, and the longest possible match is marked as a cue. For example, the classifier could output the following tag sequence:

(4) This[O] indicates[F] that[O] these[O] two[O] lethal[O] mutations[O] ...

indicates is classified as a cue but *that* is not. The list of known cues contains *'indicates that'* which matches this sentence, therefore the system prediction will be:

(5) This <indicates that> these two lethal mutations ...

Our experiments showed that the supervised system is unable to correctly detect previously unseen cues, and instead creates some unnecessary false positives. Lemma is the most important feature type for cue detection and when it is not available, there is not enough evidence to make good predictions. Therefore, we compare all system predictions to the list of known cues and if there is no match, they are removed. The detection of unseen hedge cues is a potential topic for future research.

3.5.2 Speculation scopes

The next step of our supervised approach also utilises the CRF classifier to find the scope for each predicted cue. Every word token in the sentence is tagged with either **F** (first word of a scope), **I** (inside a scope), **L** (last word of a scope) or **O** (outside, not in a scope). The correct tagging for a sample sentence is shown in Table 3.2.

	expect	**may**
We	O	O
expect	F	O
that	I	O
this	I	O
cluster	I	O
may	I	F
represent	I	I
a	I	I
novel	I	I
selenoprotein	I	I
family	L	L
.	O	O

Table 3.2: Example of scope tagging.

If a cue contains multiple words, they are each processed separately and the predictions are later combined by postprocessing rules, as described in Section 3.6.

The list of features for each token, used both alone and as sequences of 5-grams before and after the token, is:

- token-level prediction derived from the rule-based scope detection system
- lemma
- POS
- is the token also the cue
- distance from the cue
- absolute distance from the cue
- relative position to the cue
- are there brackets between the word and the cue
- is there any other punctuation between the word and the cue
- are there any special (clause separating) words between the word and cue
- is the word in the dependency subtree of the cue
- is the word in the dependency subtree of the main verb
- is the word in the dependency subject subtree of the main verb

We also include the following features of the current word in combination with the POS sequence of the cue:

- POS
- distance from the cue
- absolute distance from the cue
- relative position to the cue
- is the word in the dependency subtree of the cue
- is the word in the dependency subtree of the main verb
- is the word in the dependency subject subtree of the main verb

Additional features include:

- dependency paths between the word and the cue (full path plus subpaths with up to 5 nodes)
- dependency paths in combination with the lemma sequence of the cue

These feature sets were found by experimenting on the development data and choosing the best configuration, optimised for the F-score of the shared task evaluation.

The scope of the hedge cue can often be found by tracking the sequence of grammatical relations in the dependency graph of a sentence, as described by the manual rules. To allow the classifier to learn such regularities, we include features based on the dependency path between the cue word and the word being classified. Given that the sentence has a full parse and a connected dependency graph, we can find the shortest connected path between any two words. The path between the word and the cue is converted into a string representation, to be used as a feature value in the classifier. Path sections of different lengths allow the system to find both general and more specific patterns. POS tags are used as node values to abstract away from word tokens.

An example for the word *unusual*, using the graph from Figure 3.1, is given below. Five features representing paths with increasing lengths plus one feature containing the full path are extracted:

(6) 1: VM
2: VM<–aux–VV0
3: VM<–aux–VV0–xcomp–>VV0
4: VM<–aux–VV0–xcomp–>VV0–dobj–>NP2
5: VM<–aux–VV0–xcomp–>VV0–dobj–>NP2–ncmod–>JJ
6: VM<–aux–VV0–xcomp–>VV0–dobj–>NP2–ncmod–>JJ

Line 1 shows the POS of the cue *would* (VM). On line 2, this node is connected to *fail* (VV0) by an auxiliary dependency relation. More links are added until we reach *unusual* (JJ).

The presence of words that often indicate the beginning of a new clause, used by Morante & Daelemans (2009), is included as a feature type with values: *whereas, but, although, nevertheless, notwithstanding, however, consequently, hence, therefore, thus, instead, otherwise, alternatively, furthermore, moreover, since.*

Finally, we also experiment with combining the manual rules together with the supervised approach. First, the text is parsed and the rules from Section 3.4.2 are applied to sentences. The resulting tagging sequence is then incorporated into the CRF classifier as additional features. This method uses information from both the

manually-developed rules and automatically-extracted features, and experiments in Section 3.7 show that it achieves the highest precision and recall.

3.6 Post-processing

All of our scope prediction systems perform labelling on the token level and this does not always result in a continuous scope over the surface form of the text. Therefore, we can further increase accuracy by performing some simple heuristic post-processing steps.

First, if the cue contains multiple words, the scope predictions for each word have to be combined. This is done by overlapping the sequences and choosing the preferred tag for each word, according to the hierarchy F > L > I > O. Next, scopes are constructed from tag sequences using the following rules:

- Scope starting point is the first token tagged as F before the cue. If none are found, the first word of the cue is used instead.
- Scope end point is the last token tagged as L after the cue. If none are found, search for tags I and F. If none are found, the last word of the cue is used as end point.

The scopes are further modified until none of the rules below return any updates:

- If the last token of the scope is punctuation, move the end point before the token.
- If the last token is a closing bracket, move the scope end point before the opening bracket.
- If the last token is a number and it is not preceded by a capitalised word, move the scope end point before the token. This is a heuristic rule to handle trailing citations which are frequent in the training data and often misattached by the parser.

Finally, scopes are checked for partial overlap and any such instances are corrected. For example, the system might return a faulty version (7) of a sentence in which one scope is only partially contained within the other.

(7) We [<expect> that this cluster (<may> represent a novel] selenoprotein family).

This prediction cannot be represented using the current established format for representing hedge scopes and we were unable to find cases where such annotation would be needed. Therefore, these scopes are modified by moving the end of the first scope to the end of the second scope. The example above would become:

(8) We [<expect> that this cluster (<may> represent a novel selenoprotein family)].

3.7 Experiments

3.7.1 Dataset

Our experiments make use of the CoNLL 2010 Shared Task dataset (Farkas *et al.*, 2010), annotated for hedge cues and scopes. The training set is a revised version of the Bioscope corpus (Vincze *et al.*, 2008), which includes 9 full papers and 1,273 abstracts from biomedical research fields. A separate new set of full papers was released for evaluation as part of the shared task. Table 3.3 contains an overview of the training data statistics.

	Papers	**Abstracts**
Documents	9	1,273
Sentences	2,670	11,871
Cues	682	2,694
Scopes	668	2,659
Unique cues	100	112
Cues with multiple words	10.70%	12.25%
Scopes start with cue	72.00%	80.59%
Scopes with multiple cues	2.10%	1.28%

Table 3.3: Training data statistics.

Sentence (9) provides an example from the corpus illustrating the annotation scheme. Sentence (10) shows the same example, representing cues with angle brackets and scopes with round brackets.

(9) <sentence id="S1.166">We <xcope id="X1.166.2"><cue ref="X1.166.2" type="speculation">expect</cue> that this cluster <xcope id="X1.166.1"><cue ref="X1.166.1" type="speculation">may</cue> represent a novel selenoprotein family</xcope></xcope>.</sentence>

(10) We (<expect> that this cluster (<may> represent a novel selenoprotein family)).

The annotation guidelines (Farkas *et al.*, 2010) specify that hedge cues should be marked using the minimal strategy, where only the shortest possible unit that expresses and determines the strength of speculation should be marked as the cue. In contrast, hedge scopes should be extended to the largest syntactic unit possible. The scope of verbs, auxiliaries, adjectives and adverbs usually starts with the cue word. In the case of verbs or auxiliaries the scope includes all complements and adjuncts, ending with the clause or the sentence. The scope of predicative adjectives and sentential adverbs commonly includes the whole sentence. Other adverbs extend to the end of the clause, and attributive adjectives end with the following noun phrase. There are different exceptions to these guidelines, and linguistic phenomena, such as passive voice, can change the scope boundaries in the sentence.

There are a number of restrictions that are imposed on the annotation:

- Every cue has one scope.
- Every scope has one or more cues.
- The cue must be contained within its scope.
- Cues and scopes have to be continuous.

These limitations simplify the annotation process and the representation format, but can lead to cases where some more complex hedge scope constructions cannot be accurately annotated. Most importantly, the requirement for the hedge scope to be continuous over the surface form of a sentence does not work for some examples drawn from the training data. In (11) below it is uncertain whether

fat body disintegration is independent of the *AdoR*. In contrast, it is stated with certainty that *fat body disintegration* is *promoted by action of the hemocytes*, yet the latter assertion is included in the scope to keep it continuous.

(11) (The block of pupariation <appears> to involve signaling through the adenosine receptor (AdoR)) , but (fat body disintegration , which is promoted by action of the hemocytes , <seems> to be independent of the AdoR) .

Similarly, according to the annotation guidelines, the subject of *be likely* should be included in its scope, as shown in example (12). In sentence (13), however, the subject *protein pairs* is separated by two non-speculative clauses and is therefore excluded from the scope.

(12) Some predictors make use of the observation that (neighboring genes whose relative location is conserved across several prokaryotic organisms are <likely> to interact).

(13) Protein pairs predicted to have a posterior odds ratio below 1.0 have an estimated true positive rate below 50% and thus (are more <likely> not to interact than to interact).

In (14), arguably, there is no hedging as the sentence precisely describes a statistical technique for predicting interaction given an assumption.

(14) More recently, other groups have come up with sophisticated statistical methods to estimate (<putatively> interacting domain pairs), based on the (<assumption> of domain reusability).

These examples illustrate several cases where the current guidelines produce controversial or unintuitive annotation, which in turn can lead to low agreement of human annotators and low upper bounds for automated systems.[1] Ultimately, dealing effectively with such examples would involve representing hedge scopes

[1]We are unable to report annotator agreement on the dataset, as it has not been cross-annotated.

in terms of sets of semantic propositions recovered from a logical semantic representation of the text, in which anaphora, word sense, and entailments had been resolved. However, the currently established format can correctly represent a large majority of scopes and reasonably approximate the remaining cases. It is part of future work to investigate whether a more accurate annotation scheme can be created for the representation of hedge scopes.

All of the training and test sentences were tokenised and parsed using the RASP system (Briscoe *et al.*, 2006). Multiple part-of-speech (POS) tag outputs were passed to the parser (to compensate for the high number of unseen words in biomedical text), retaining just the highest ranked directed graph of dependency relations. Each node in the graph represents a word token annotated with POS, lemma, and positional order information. In the case of parse failure the set of unconnected graphs returned by the highest-ranked spanning subanalyses for each sentence was retained.

3.7.2 Cue detection

We make use of the evaluation scripts provided by the organisers of the shared task. In cue-level evaluation a predicted cue is counted as correct if it contains the correct substring of the sentence. Token-level evaluation would not give accurate results because of varying tokenisation rules. The sentence-level evaluation regards a sentence as being speculative if it contains one or more cues. We report precision, recall and F-measure of different system configurations using both evaluation methods.

As the baseline system for cue detection we use simple string matching. The list of known cues is collected from the training data and compared to the test sentences. The longest possible match is always marked as a cue.

The rule-based system uses a limited list of 42 word sequences that predict speculation with higher precision. The supervised system uses a fully-supervised classification system to predict a cue label for each token. During the development process we separated the training set into two sections and used 60% for training and 40% for development. The results below measure the system performance on the evaluation dataset while using all of the training data to build the supervised

classifier.

	Baseline	**Rule-based**	**Supervised**
Predicted cues	3062	889	991
Correct cues	1018	737	808
Cue precision	33.25	82.90	81.53
Cue recall	97.23	70.39	77.17
Cue F-measure	49.55	76.14	79.29
Sentence precision	41.32	85.81	83.82
Sentence recall	99.49	79.62	83.92
Sentence F-measure	58.40	82.60	83.87

Table 3.4: Cue detection results on the evaluation dataset.

The baseline system returns nearly all cues but since it matches every string, it also returns many false positives, resulting in low precision and only 49.55% F-measure. The rule-based system uses a carefully selected list of cue words and increases F-measure to 76.14%. The supervised system further improves recall while sacrificing some precision and manages to achieve the highest F-measure of 79.29%. It is worth noting that the difference between the last two systems is only 3.15%, despite of the vast differences in complexity. The difference in sentence-level performance is even smaller – 82.60% F-measure from the rule-based system, compared to 83.87% from the supervised system. Based on these experiments we believe that a minimally-supervised method could be a practical choice for production-level deployment, achieving comparable results with virtually no complexity or extra resources. However, when trying to achieve the best possible performance, supervised classifiers can be used to model much more complicated patterns and reach the highest F-measure.

In the context of the shared task, our supervised system ranked fourth in the cue-level evaluation, with the top system achieving 81.3% F-measure. If the rule-based method had also taken part, it would have ranked 9th out of 16 participating systems.

For further analysis of the cue detection task, Table 3.5 lists the ten most common cues in the test data and the number of cues found by the supervised system. In the cases of *may* and *suggest*, which are also the most common cues

	TP	FP	Gold
may	161	5	161
suggest	124	0	124
can	2	1	61
or	9	12	52
indicate that	49	2	50
whether	42	6	42
might	42	1	42
could	30	17	41
would	37	14	37
appear	31	14	31

Table 3.5: True and false positives of the ten most common cues in the evaluation data, using the best supervised system.

in the development data, the system finds all the correct instances. *Can* and *or* are not detected as accurately because they are both common words that in most cases are not functioning as speculation cues. For example, there are 1,215 occurrences of *or* in the training data and only 146 of them are hedge cues; *can* is a cue in 64 out of 506 instances. We have not found any extractable features that reliably distinguish between the different uses of these words.

3.7.3 Scope detection

A scope is counted as correct if it has the correct beginning and end locations in the sentence, and is associated with the correct cues. Scope prediction systems require detected cues as input, therefore we present three separate evaluations – using correct cues from the gold standard, using cues predicted by the rule-based system (Section 3.4.1), and using cues predicted by the supervised system (Section 3.5.1).

The baseline method looks at each cue and marks a scope from the beginning of the cue to the end of the sentence, excluding the ending punctuation. The system using manual rules applies the appropriate rule, based on the POS tag of the cue, and selects words that belong to the scope, as described in Section 3.4.2. The supervised system uses a CRF classifier to label tokens based on a range of

features, such as the lemma, part-of-speech, surrounding words, relative position to the cue, and dependency path to the cue. The combined system takes the labels from the rule-based predictions and uses them as additional features in the supervised system. Post-processing rules are applied equally to the output of all methods.

	# correct	Precision	Recall	F-measure
Baseline	596	56.92	57.70	57.31
Rule-based	661	63.86	63.99	63.93
Supervised	633	60.46	61.28	60.87
Combined	683	65.99	66.12	66.05

Table 3.6: Scope detection results using the gold standard cues.

	# correct	Precision	Recall	F-measure
Baseline	470	52.87	45.50	48.91
Rule-based	493	55.46	47.73	51.30
Supervised	498	56.02	48.21	51.82
Combined	526	59.17	50.92	54.73

Table 3.7: Scope detection results using cues found by the rule-based system.

	# correct	Precision	Recall	F-measure
Baseline	506	51.06	48.98	50.00
Rule-based	532	53.74	51.50	52.60
Supervised	537	54.19	51.98	53.06
Combined	565	57.07	54.70	55.86

Table 3.8: Scope detection results using cues found by the supervised system.

Tables 3.6, 3.7 and 3.8 present the experimental results using different methods for cue detection. The baseline system performs remarkably well compared to other approaches. It does not use any grammatical or lexical knowledge apart from the location of the cue and yet it delivers an F-score of 50.00% with predicted and 57.31% with gold standard cues.

Manual rules are essentially a more fine-grained version of the baseline. Instead of a single rule, one of 8 possible rules is selected, based on the POS tag of the cue. This improves the results, increasing the F-score to 52.60% with predicted and 63.93% with gold standard cues. The improvement suggests that the POS tag of a cue is a useful indicator of how it behaves in a sentence.

The supervised system gives a small additional improvement when using predicted cues, but it is outperformed by the rule-based system if gold-standard cues are provided as input. This can be explained by the classifier learning correct patterns for common cues that are also frequent in the training data, but performing poorly on rare or unseen cues. Since the cue prediction systems are also better at predicting common cues, the supervised classifier has an advantage given this specific input. However, when dealing with cues of lower frequency, the linguistically-engineered scope detection rules perform better. A possible avenue for future work is to investigate if using a different scope prediction system based on the frequency of the cue could further improve accuracy.

Finally, the combination system uses information from both sources and achieves the best overall results, with 66.05% F-measure using gold-standard cues and 55.86% using predicted cues. This configuration ranked second in the scope-level evaluation of the CoNLL-10 Shared Task; in comparison, the best-performing system by Morante *et al.* (2010) achieved 57.3% F-measure with predicted cues. The system using only heuristic rules for both cue and scope detection would have ranked 6th out of 15 participants with 51.30% F-measure. Furthermore, the simple method of detecting cues using a list of 42 phrases and always constructing the scope from the cue to the end of the sentence achieves 48.91% F-measure and would have also ranked in the 6th place.

Error analysis on the output of the rule-based system showed that 35% of faulty scopes were due to incorrect or unconnected dependency graphs output by the parser, and 65% due to exceptions that the manual rules do not cover. An example of an exception is given by Sentence 15, with the braces { } showing the scopes predicted by the rules.

(15) Contamination is {(<probably> below 1%)}, which is {(<likely> lower than the contamination rate of the positive dataset) as discussed in 47}.

In this sentence, *as discussed in 47* is a modifier of the clause which is usually included in the scope, but in this case should be left out.

3.8 Conclusion

This chapter investigated the task of detecting speculation cues and scopes using systems with varying levels of supervision. We found that the dependency output from the RASP system can be effectively used as features in a supervised classifier for detecting cues, and also as the basis for manual rules and features for scope detection. We demonstrated that a small number of manual rules can provide competitive results, and that these can be further improved using machine learning techniques and post-processing rules. The generally low results for the scope detection task demonstrate the difficulty of both annotating and detecting the hedge scopes in terms of surface sentential forms. A revised definition of the overall task could lead to better systems, as the current guidelines can seem unintuitive in certain cases.

We compared the performance of a minimally supervised rule-based system to a fully supervised system and found that the small number of manual rules were able to offer competitive results without requiring any annotation. Furthermore, this requires much less resources and can therefore be more suitable for practical deployment. However, for maximum performance, fully supervised methods offer the way to learn more complicated patterns in the data and achieve the highest accuracy. Combining the alternative methods together improves the results even further.

There are several interesting directions for further investigating the problem of speculation detection. We would encourage further development of manual rules for scope resolution, as more fine-grained approaches could potentially lead to performance comparable to supervised systems. Also, Morante *et al.* (2010), who ranked first in the task of scope detection, used a k-nearest neighbour algorithm instead of a probabilistic approach. Using the features developed for our classifier together with a k-NN approach could give interesting results.

The CoNLL-10 Shared Task triggered the creation of dozens of hedge detection systems. While their evaluation scores are comparable on the shared dataset, it

is currently unclear which components contribute most to achieving the best results. Different implementations of cue and scope detection systems make use of a wide range of manual rules, classifiers, parsers, features, post-processing steps and external resources. Removing some of these variables and conducting experiments on specific components could also lead to the development of better systems.

Haghighi *et al.* (2005) point out three types of language constructions that make the task of textual entailment recognition more difficult – hedging, negation and superlative adjectives. These language constructions can change the direction of an entailment relation, yet they often go unnoticed by simple graph matching techniques. Our experiments in Chapter 5 also demonstrate that detection of speculative and negated language can directly improve the performance of entailment detection systems. Negation and superlative adjectives behave similarly to speculation – there exist certain cue words that have influence over specific parts of the sentence. As we have a working system for speculation scope detection, an interesting future extension would be to adapt this technique to also work with cases involving negation and superlative adjectives.

Chapter 4

Hyponym detection and generation

4.1 Introduction

Hyponymy is a relation between two word senses, indicating that the meaning of one word is also contained in the other. It can be thought of as a *type-of* relation; for example *car*, *ship* and *train* are all hyponyms of *vehicle*. We indicate a hyponymy relation between words a and b as $(a \rightarrow b)$, showing that a is a hyponym of b, and b is a hypernym of a. For example, $(ship \rightarrow vehicle)$ and $(villa \rightarrow house)$. Hyponymy relations are closely related to the concept of entailment, and this notation is consistent with indicating the direction of inference – if a is true, b must be true as well.

Automatic detection and generation of hyponyms has many practical applications in nearly all natural language processing tasks. Information retrieval, information extraction and question answering can be improved by performing appropriate query expansion. For example, a user searching for *arthritis treatment* is most likely also interested in results containing the hyponyms of *treatment*, such as *arthritis therapy*, *arthritis medication*, and *arthritis rehabilitation*. Summarisation systems can increase coherence and reduce repetition by correctly handling hyponymous words in the input text. Entailment and inference systems can improve sentence-level entailment resolution by detecting the presence and

direction of word-level hyponymy relations. Distributionally similar words have been used for smoothing language models and word co-occurrence probabilities (Dagan *et al.*, 1999; Weeds & Weir, 2005), and hyponyms are potentially more suitable for this task. For example, we may wish to estimate the probability of *publication* being the object of *read*, but due to sparsity this combination never appears in our corpus. By finding that *book* is a hyponym of *publication*, and that *book* occurs with *read* quite frequently, we can infer that publications are also likely to be read.

Most previous work has been using the general notion of distributional or semantic similarity between two words. Systems often employ symmetric similarity scores, potentially leading to incorrect behaviour. For example, *vehicle* can be replaced by *train* in a query expansion setting, since every train is also a vehicle and therefore matches the original query. However, replacing *train* with *vehicle* would likely return incorrect results, as it would also refer to many other types of vehicles. In addition, different definitions of semantic similarity or relatedness cover a wide range of relations, some of which are not suitable for word substitution. For example, *window* is a meronym of *house*, but although *brick house* is a likely co-occurrence, *brick window* is an unlikely word pair and does not semantically match the original phrase. Therefore, we focus on hyponymy relations, as they are asymmetrical and better suited for the applications of query expansion and lexical substitution.

The task of **hyponym detection** is to determine whether one word is a hyponym of the other. Given a directional word pair (a, b), the system needs to classify it as a valid hyponym relation $(a \rightarrow b)$ or not. This can be useful when the application has a list of candidate pairs, or when looking for similarities between two sentences. In contrast, **hyponym acquisition** is the task of extracting all possible hyponym relations from a given resource or text corpus. Such systems often make use of heuristic rules and patterns for extracting relations from surface text, and populate a database with hyponymous word pairs. Finally, the task of **hyponym generation** is to return a list of all possible hyponyms, given only a single word as input. This is most relevant to practical applications, as many systems require a set of appropriate substitutes for a specific term, yet barely any research has been done in this area. The problem of generation can be

reformulated as a hyponym detection task by using a very large set of candidate hyponyms, or as a hyponym acquisition task by utilising a large corpus containing all the required word pairs. However, all successful methods for hyponym detection or acquisition are not necessarily well-suited for hyponym generation, as it involves a different range of challenges.

Our work focuses on the task of hyponym generation and approaches it through hyponym detection. We are especially interested in methods using distributional similarity, as they can be easily deployed on different domains and genres without requiring annotated training data. First, we created several new datasets that help us evaluate the performance of different measures on these tasks. Then, we performed systematic assessment of a wide range of distributional similarity measures, which are prominent in existing research, on the tasks of hyponym detection and generation. A novel measure is proposed that builds on previous work and delivers significantly better results on our dataset of noun hyponym generation. Finally, we examine how good is the performance of unsupervised similarity measures when compared to a fully supervised approach using annotated data. A discriminative SVM classifier was trained using additional labelled examples and used to run comparative experiments on the same problem sets.

4.2 Research goals

Our work on hyponym detection and generation has the following goals:

- Construct appropriate datasets for evaluating hyponym detection and generation.
- Compare the performance of well-known similarity measures on the tasks of hyponym detection and generation.
- Investigate the properties of a successful directional similarity measure.
- Specify a new similarity measure for hyponym generation.
- Compare the performance of minimally supervised distributional similarity measures to a fully supervised machine learning algorithm.

4.3 Background

Hearst (1992) described one of the earliest methods for automated hyponym acquisition from corpora. Several lexico-syntactic patterns were manually defined to model common contexts in which hyponyms appear. For example, pattern (16) matches sentence (17):

(16) such NP as {NP,}* {(or |and)} NP

(17) ... works by such authors as Herrick, Goldsmith and Shakespeare.

Their system is low-cost and applicable even to very small text corpora, but it relies heavily on very specific lexical patterns matching the input text, and is unable to offer sufficiently wide coverage. Snow *et al.* (2005) extend this idea by having the system automatically learn useful lexico-syntactic patterns using an annotated dataset.

Various alternative approaches have been developed and applied to the related task of automated ontology creation. Distributional similarity measures have received wide attention as well, but mostly as methods for hierarchical clustering when building prototype-based ontologies (Ushioda, 1996; Bisson *et al.*, 2000; Wagner, 2000; Paaß *et al.*, 2004). In a prototypical ontology, every node contains a set of terms that are found to be sufficiently similar, but the structure does not contain the labels necessary for performing lexical substitution or inducing hyponym relations. Manually constructed patterns, based on the method proposed by Hearst (1992), can be used to assign suitable labels to each of the clusters in a post-processing stage (Caraballo, 1999; Cimiano & Staab, 2005).

Discovering a hierarchy between concepts can also be formulated as a supervised classification task, given appropriate training data. In such a setting, an existing ontology can be used for feature extraction and evaluation, and new relations returned by the system can be suggested as potential additions (Alfonseca & Manandhar, 2002; Witschel, 2005; Snow *et al.*, 2006). Recently, more advanced methods using Markov Logic have been deployed to jointly induce a probabilistic ontology and perform knowledge extraction (Poon & Domingos, 2009, 2010).

Biemann (2005) offers an overview of different tasks and methods related to automated ontology creation. However, very limited work has been directed towards using distributional methods for hyponym detection, and to the best of our knowledge there has been no previous work attempting to evaluate these methods for hyponym generation.

Zhitomirsky-Geffet & Dagan (2009) focus on improving feature vector quality through bootstrapping, and apply it to the task of detecting meaning-preserving words for lexical substitution. They define the relation of **substitutable lexical entailment** (or **lexical entailment** for brevity). Given a directional pair of terms (w, v), the term w is said to entail v if the following two conditions are fulfilled:

1. Word meaning entailment: the meaning of a possible sense of w implies a possible sense of v.

2. Substitutability: w can substitute for v in some naturally occurring sentence, such that the meaning of the modified sentence would entail the meaning of the original.

As an example, *government* entails *minister*, since *The government voted for the new law* entails *A minister in the government voted for the new law*.

The substitutability criterion requires there to be one context where v can be substituted by w for there to exist a valid entailment relation. However, the simple notion of substitutability can in some cases be too general, leading to ambiguous results. From the previous example, *minister* should also entail *government*, as *Ministers passed a new law* entails *The government passed a new law*. Similarly, *He likes fruit* entails *He likes apples*, but *He eats apples for breakfast* entails *He eats fruit for breakfast*. This example seems to indicate that *apple* and *fruit* are interchangeable and essentially synonymous. Other linguistic phenomena such as negation, hedging and quantifiers can further modify the direction of the entailment relation, and without additional criteria it can be difficult to unambiguously determine lexical entailment. In this chapter, we have chosen to focus on detecting only hyponymy relations, as they are defined between specific concepts, regardless of the context where they appear, and still cover

the majority of word pairs that are suitable for lexical substitution in practical applications.

Our work follows Kotlerman *et al.* (2010), who offer one of the most recent comparisons of different distributional similarity measures for detecting lexical entailment. They describe the theoretical criteria of a good directional similarity measure, and also propose a new measure based on average precision. We apply similar methodology to hyponym detection and generation, further extend their work by suggesting additional desired properties of a directional similarity measure, and describe a new measure which delivers significant improvements in our evaluation. As hyponymy is dependent on word senses, we also follow their approach and define a relation to be valid between two terms if it is valid between any senses of those terms.

Recently, Baroni & Lenci (2011) created the BLESS data set for evaluation of distributional semantic models. It contains annotated tuples with various relations, including hypernyms, but it is designed for the task of distinguishing different types of alternative relations, as opposed to generating hyponyms from an open candidate set.

4.4 Similarity measures

In this section we present an overview of prominent distributional similarity measures which have been used to detect semantic similarity and which could also be used for hyponymy detection. F_a is used to denote the set of weighted features for word a, $f \in F_a$ indicates one feature instance present in that set, and $w_a(f)$ is the weight of feature f for word a. Examples of possible feature types include words that occur in a limited context window, connected words and relations from dependency graphs, or document-level co-occurrences. Since the possible feature space can be infinite, we say that f belongs to the feature set F_a if it has a non-zero weight. Some measures require only positive feature weights to return reasonable similarity scores, and we use F_a^+ to indicate the subset of features with strictly positive values. Our implementations always return the similarity value of 0 if they cannot be calculated – for example, when the input of empty vectors would lead to division by zero.

Most of these measures are true similarities, assigning higher score values to more similar word pairs, usually in the range of [0,1] or [-1,1]. However, there are also some distance measures, with higher values showing higher dissimilarity, and they are indicated by a diamond ($\diamond$) as their output needs to be reverse-ordered. Similarly, we also use a star ($\star$) to indicate measures that are asymmetric – the similarity score for (a, b) will not necessarily have the same value as for (b, a).

4.4.1 Cosine

Cosine similarity calculates the angle between two feature vectors and has become the standard measure for similarity between weighted vectors in information retrieval (IR). Using the inner product between two vectors as a similarity score developed naturally by generalising coordinate-wise matching. However, inner product by itself does not take vector length into account and will assign higher values to longer vectors. In contrast, simply finding the Euclidean distance between two vectors strongly discriminates against vectors with different lengths. The cosine measure overcomes both of these problems by considering the direction of the two vectors, as opposed to their length or distance (Witten *et al.*, 1999; Curran, 2003):

$$Cosine(F_a, F_b) = \frac{\sum_{f \in F_a \cap F_b} w_a(f) \times w_b(f)}{\sqrt{\sum_{f \in F_a} w_a(f)^2} \times \sqrt{\sum_{f \in F_b} w_b(f)^2}} \tag{4.1}$$

4.4.2 Pearson product-moment correlation coefficient

The Pearson product-moment correlation coefficient, typically denoted by r, was defined by Karl Pearson in 1895 (Pearson, 1895; Rodgers & Nicewander, 1988) and it measures correlation between two variables. The value is in the range of [-1,1], with 0 denoting no correlation and -1 meaning negative correlation. The formula is presented below:

$$r(F_a, F_b) = \frac{\sum_{f \in F_a \cup F_b} [(w_a(f) - \mu_a) \times (w_b(f) - \mu_b)]}{\sqrt{\sum_{f \in F_a \cup F_b} (w_a(f) - \mu_a)^2} \times \sqrt{\sum_{f \in F_a \cup F_b} (w_b(f) - \mu_b)^2}} \tag{4.2}$$

where μ_a is the mean of feature weights for vector F_a:

$$\mu_a = \frac{1}{|F_a \cup F_b|} \sum_{f \in F_a \cup F_b} w_a(f) \tag{4.3}$$

where $|F_a \cup F_b|$ is the number of unique elements in both vectors combined.

When calculating Pearson's or Spearman's correlation, it is important to note that zero values cannot be simply ignored, as in cosine similarity. Here, the mean of feature weights is subtracted from each weight, giving these elements a non-zero value. Therefore, the calculation has to be done over the union of both sets of features.

One interpretation of this measure is that both feature vectors are modified by subtracting the corresponding mean of the weights. The cosine similarity between these modified vectors is also equal to Pearson's coefficient.

4.4.3 Spearman's rank correlation coefficient

Spearman's correlation coefficient (Spearman, 1904), denoted by ρ, is defined as the Pearson product-moment correlation coefficient between the ranks of feature weights:

$$\rho(F_a, F_b) = \frac{\sum_{f \in F_a \cup F_b} [(r_a(f) - \mu'_a) \times (r_b(f) - \mu'_b)]}{\sqrt{\sum_{f \in F_a \cup F_b} (r_a(f) - \mu'_a)^2} \times \sqrt{\sum_{f \in F_a \cup F_b} (r_b(f) - \mu'_b)^2}} \tag{4.4}$$

where $r_a(f)$ is the rank of feature f in the sorted order of vector F_a, and μ'_a is the mean of rank values. In case of tied values in the feature vector, the rank is calculated as the average of the ranks. For example, if identical values are positioned in ranks 2 and 3, they are both assigned the rank value of 2.5. As with Pearson's correlation, Spearman's correlation is also symmetrical and with values in the range of -1 to 1. However, unlike Pearson's, this measure is less sensitive to outliers, as the specific weights are reduced to only relative positions.

4.4.4 Jaccard index (Set)

Jaccard index (Jaccard, 1901) is a statistic used for comparing the similarity between two sets. It is calculated as the size of the intersection of these sets, divided by the size of the set union. Jaccard index can be used for measuring the similarity of two feature vectors by applying it to the sets of non-zero feature types. This method ignores the specific feature weights and treats them as binary values:

$$JaccardSet(F_a, F_b) = \frac{|F_a \cap F_b|}{|F_a \cup F_b|} \tag{4.5}$$

4.4.5 Dice (Set)

Dice's coefficient, created by Dice (1945), is similar to the Jaccard index and operates over sets. It compares the size of the intersection to the sum of individual set sizes. As before, we can apply this to feature vectors by operating over sets of feature types with non-zero weights:

$$DiceSet(F_a, F_b) = \frac{2 \times |F_a \cap F_b|}{|F_a| + |F_b|} \tag{4.6}$$

4.4.6 Overlap coefficient

The overlap coefficient (Manning & Schütze, 1999) is similar to the set similarity measures of Jaccard and Dice. However, it compares the size of the intersection only to the smaller of the two sets:

$$OverlapSet(F_a, F_b) = \frac{|F_a \cap F_b|}{min(|F_a|, |F_b|)} \tag{4.7}$$

4.4.7 Cosine (Set)

This early set-theoretic version of cosine is similar to DiceSet, except the denominator uses a geometric mean instead of an arithmetic mean. They are identical when both vectors contain the same number of non-zero elements. However, the CosineSet measure gives a smaller penalty to cases where the number of non-zero elements is very different. For example, when comparing vectors of length 1

and length 1000, with one shared entry, then $CosineSet = 1/\sqrt{1000 \times 1} \approx 0.03$, whereas $DiceSet = 2 \times 1/(1 + 1000) \approx 0.0002$. This is a useful property for statistical NLP, since we often compare words with varying amounts of features or evidence, but that does not necessarily mean they are not similar in semantics (Manning & Schütze, 1999).

$$CosineSet(F_a, F_b) = \frac{|F_a \cap F_b|}{\sqrt{|F_a| \times |F_b|}} \tag{4.8}$$

4.4.8 Jaccard (Generalisation)

Grefenstette (1994) generalised the Jaccard similarity measure to non-binary value (fuzzy sets) semantics. In this framework, the intersection operation is replaced with the sum of minimum pairwise feature weights, and the union is replaced with the sum of maximum pairwise feature weights. If the weights are constrained to binary values, this measure reduces to the original Jaccard index performed on sets.

$$JaccardGen(F_a, F_b) = \frac{\sum_{f \in F_a \cap F_b} min(w_a(f), w_b(f))}{\sum_{f \in F_a \cup F_b} max(w_a(f), w_b(f))} \tag{4.9}$$

4.4.9 Dice (Generalisation)

Using a similar approach, the Dice similarity measure can also be generalised to weighted feature vectors (Grefenstette, 1994; Curran, 2003):

$$DiceGen(F_a, F_b) = \frac{2 \times \sum_{f \in F_a \cap F_b} min(w_a(f), w_b(f))}{\sum_{f \in F_a} w_a(f) + \sum_{f \in F_b} w_b(f)} \tag{4.10}$$

4.4.10 Dice (Generalisation 2)

Curran (2003) describes an alternative way of generalising the Dice measure to real-valued weights, based on Grefenstette (1994). Instead of finding the minimum of two weights, the values are multiplied together. It is pointed out that the same formula can also be considered as an alternative generalisation for the

Jaccard measure:

$$DiceGen2(F_a, F_b) = \frac{\sum_{f \in F_a \cap F_b} w_a(f) \times w_b(f)}{\sum_{f \in F_a} w_a(f) + \sum_{f \in F_b} w_b(f)} \tag{4.11}$$

4.4.11 Kendall's tau coefficient

Kendall's tau, denoted by τ, is a non-parametric statistic used to measure the rank correlation between random variables (Kendall, 1938; Kruskal, 1958). It counts the number of pairwise items that are in the same relative position in both distributions. The Tau-b variation of the measure also takes into account the number of possible tied values:

$$\tau(F_a, F_b) = \frac{\sum_{f,g \in F_a \cup F_b} sign((w_a(f) - w_a(g)) \times (w_b(f) - w_b(g)))}{\sqrt{(n_0 - n_a) \times (n_0 - n_b)}} \tag{4.12}$$

where n_0 is the total number of pairs:

$$n_0 = |F_a \cup F_b| \times (|F_a \cup F_b| - 1) \tag{4.13}$$

and n_a is the total number of tied pairwise values in set a:

$$n_a = \sum_i t_i^a (t_i^a - 1)/2 \tag{4.14}$$

where t_i^a is the number of tied values in the ith group of ties for set a.

While both Spearman's ρ and Kendall's τ measure correlation between rankings, Kendall's τ penalises dislocations based on the linear distance of the dislocation, whereas Spearman's ρ uses the square of the distance.

4.4.12 Lin similarity

Lin similarity was created by Lin (1998), and uses the ratio of shared feature weights compared to all feature weights.

$$Lin(F_a, F_b) = \frac{\sum_{f \in F_a^+ \cap F_b^+} [w_a(f) + w_b(f)]}{\sum_{f \in F_a^+} w_a(f) + \sum_{f \in F_b^+} w_b(f)} \tag{4.15}$$

where F_a^+ denotes the subset of features in F_a that have positive weights. This measure finds a weighted proportion of features that are shared by both words. The calculation is done only over positive values, as including negative weights would return an incorrect ratio.

4.4.13 Weeds' Precision ⋆

Weeds *et al.* (2004) proposed using precision and recall as a measure of similarity. In this framework, the features are treated similarly to retrieved documents in information retrieval – the vector of the broader term is used as the gold standard, and the vector of the narrower term is in the role of retrieval results. Precision is then calculated by comparing the intersection (*items correctly returned*) to the values of the narrower term only (*all items returned*).

$$WeedsPrec(F_a, F_b) = \frac{\sum_{f \in F_a^+ \cap F_b^+} w_a(f)}{\sum_{f \in F_a^+} w_a(f)} \tag{4.16}$$

where F_a^+ refers to the subset of features in F_a that have positive non-zero weights.

4.4.14 Weeds' Recall ⋆

Similar to precision, recall can also be formulated as a similarity measure (Weeds *et al.*, 2004). In this case the calculation is done over the weights in the broader term, quantifying how well the narrower term encompasses the broader one:

$$WeedsRec(F_a, F_b) = \frac{\sum_{f \in F_a^+ \cap F_b^+} w_b(f)}{\sum_{f \in F_b^+} w_b(f)} \tag{4.17}$$

4.4.15 Weeds' F-score

Following IR evaluation methods, the scores for precision and recall can be combined into a single measure by finding their harmonic mean (Weeds *et al.*, 2004):

$$WeedsF(F_a, F_b) = \frac{2 \times WeedsPrec(F_a, F_b) \times WeedsRec(F_a, F_b)}{WeedsPrec(F_a, F_b) + WeedsRec(F_a, F_b)} \tag{4.18}$$

4.4.16 Clarke's degree of entailment ⋆

Clarke (2009) proposed the asymmetric *degree of entailment* measure, based on the concept of *distributional generality* (Weeds *et al.*, 2004). It quantifies the weighted coverage of the features of the narrower term a by the features of the broader term b:

$$ClarkeDE(F_a, F_b) = \frac{\sum_{f \in F_a \cap F_b} min(w_a(f), w_b(f))}{\sum_{f \in F_a} w_a(f)} \tag{4.19}$$

4.4.17 Average precision ⋆

Kotlerman *et al.* (2010) proposed using a modified version of average precision (AP), often used as an IR evaluation metric, as a directional similarity measure instead. In this framework, the features of the broader term are analogous to the set of all relevant documents, whereas the features of the narrower term correspond to retrieved documents. The similarity score will be greater if a large number of features occur in both vectors. However, it also takes into account the ranking of these features, and gives greater importance to highly ranked features from the narrower term a:

$$AP(F_a, F_b) = \frac{\sum_{r=1}^{|F_a|}[P(r) \times rel(f_r)]}{|F_b|} \tag{4.20}$$

where

$$P(r) = \frac{|f \in F_a(r) \cap F_b|}{r} \tag{4.21}$$

$$rel(f) = \begin{cases} 1 & \text{if } f \in F_b \\ 0 & \text{if } f \notin F_b \end{cases} \tag{4.22}$$

$F_a(r)$ is used to denote the subset of features from F_a that are ranked 1 to r after sorting in descending order, placing the highest-weighted feature at rank 1. $P(r)$ calculates precision at rank r – the number of features that are in ranks 1 to r in F_a and also present in F_b, normalised by the rank r. $rel(f_r)$ takes the feature that is ranked r in F_a, returning 1 if it is also present in F_b, and 0 otherwise.

4.4.18 Average precision (inclusion) ⋆

Kotlerman *et al.* (2010) also further modify the Average Precision measure to better model directional similarity between words. First, they use the number of features in F_a for normalisation, instead of F_b. This finds the proportion of shared features relative to the features for the narrower term, better capturing their desired property of feature inclusion. In addition, they redefine the relevance function to give more fine-grained values depending on where the shared feature ranks in the list of ordered features for F_b, assigning higher weights to higher ranks:

$$APInc(F_a, F_b) = \frac{\sum_{r=1}^{|F_a|}[P(r) \times rel'(f_r)]}{|F_a|} \tag{4.23}$$

$$rel'(f) = \begin{cases} 1 - \frac{r_b(f)}{|F_b|+1} & \text{if } f \in F_b \\ 0 & \text{if } f \notin F_b \end{cases} \tag{4.24}$$

where $r_b(f)$ is used to denote the rank of feature f in the list of features in F_b, ranked by weight in descending order. The maximum value of this measure is 0.5, as the $rel'(f)$ function penalises lower ranks, even when in perfect order.

4.4.19 Average precision (balanced inclusion) ⋆

Finally, Kotlerman *et al.* (2010) combine the APInc measure with the Lin similarity (Lin, 1998) by taking their geometric average. Since APInc models the similarity using the features of the narrower term F_a, it can give unreliable re-

sults when the number of features $|F_a|$ is very small. The motivation behind combining these measures is that the symmetric Lin measure will decrease the final score for such word pairs, thereby balancing the results:

$$BalAPInc(F_a, F_b) = \sqrt{Lin(F_a, F_b) \times APInc(F_a, F_b)} \tag{4.25}$$

4.4.20 Directional Lin ⋆

We proposed an asymmetric version of the Lin measure, and used it for entailment detection between text fragments (Rei & Briscoe, 2011). Features of word b which are not found in word a are excluded from the calculation:

$$LinD(F_a, F_b) = \frac{\sum_{f \in F_a^+ \cap F_b^+} [w_a(f) + w_b(f)]}{\sum_{f \in F_a^+} w_a(f) + \sum_{f \in F_a^+ \cap F_b^+} w_b(f)} \tag{4.26}$$

4.4.21 Balanced precision ⋆

Szpektor & Dagan (2008) proposed combining WeedsPrec together with the Lin measure by taking their geometric average. This aims to balance the WeedsPrec score, as the Lin measure will penalise cases where one vector contains very few features:

$$BalPrec(F_a, F_b) = \sqrt{Lin(F_a, F_b) \times WeedsPrec(F_a, F_b)} \tag{4.27}$$

4.4.22 Kullback-Leibler divergence ◇ ⋆

The relative entropy of two probability mass functions $p(x)$ and $q(x)$ is given by:

$$D(p||q) = \sum_i p_i log \frac{p_i}{q_i} \tag{4.28}$$

where we define $0 \times log(\frac{0}{q}) = 0$. It is also known as the Kullback-Leibler divergence (Kullback & Leibler, 1951; Kullback, 1959) and measures the difference of two probability distributions over the same space. Intuitively it measures how well the *true* distribution p is modelled by the approximation q. This quantity is non-symmetric ($P(p||q) \neq P(q||p)$) and equals 0 iff $p = q$.

It is originally defined only over valid probability distributions, but we redefine it as a measure between feature vectors. Given that the feature weights correspond to the values in the probability distribution, the final score will be the same. Since the result is ∞ whenever there is at least one dimension where $q_i = 0$ and $p_i \neq 0$, we define the Kullback-Leibler divergence only over features that have positive weights in both vectors:

$$KLDivergence(F_a, F_b) = \sum_{f \in F_a^+ \cap F_b^+} w_b(f) log_e(\frac{w_b(f)}{w_a(f)}) \tag{4.29}$$

As Weeds (2003) point out when discussing the highly-related α-skew measure (see Section 4.4.24), it can be non-trivial to choose the direction when applying these measures. While the formula above matches the original description and is more theoretically-justified, reversing the arguments can lead to better performance for the α-skew. Therefore, we also report the reversed version of KL-Divergence, calculating how well the narrower term is modelled by the broader term:

$$KLDivergenceR(F_a, F_b) = \sum_{f \in F_a^+ \cap F_b^+} w_a(f) log_e(\frac{w_a(f)}{w_b(f)}) \tag{4.30}$$

4.4.23 Jensen-Shannon divergence ◇

Jensen-Shannon divergence, also known as 'information radius' (IRad) (Manning & Schütze, 1999) or 'total divergence to the average' (Dagan *et al.*, 1997), is a way of overcoming the division-by-zero issue in KL-Divergence and also making it into a symmetric measure. It is defined through KL-Divergence and intuitively answers the question of how much information is lost if we describe the two words using only their average distribution:

$$A(p, q) = D(p||\frac{p+q}{2}) + D(q||\frac{p+q}{2}) \tag{4.31}$$

$$JSDivergence(F_a, F_b) = \sum_{f \in F_a^+ \cup F_b^+} [[w_a(f) \times log_e \frac{w_a(f)}{\frac{w_a(f)+w_b(f)}{2}}] + [w_b(f) \times log_e \frac{w_b(f)}{\frac{w_a(f)+w_b(f)}{2}}]] \tag{4.32}$$

4.4.24 α-skew divergence $\diamond$ $\star$

Lee (1999) observed that symmetric measures were unable to model substitutability of words in context, and created an alternative directional measure based on KL-divergence. Since KL-Divergence will return infinity whenever $q_i = 0$ and $p_i \neq 0$, it was proposed to smooth the distribution of q using the distribution of p, therefore making sure that the reference distribution has always non-zero values:

$$S_\alpha(p, q) = D(q||\alpha p + (1 - \alpha)q) \tag{4.33}$$

The measure includes parameter α to control the degree to which the distribution of q is used to smooth p. α-skew can be viewed as an approximation of KL-divergence and the two measures are equal if $\alpha = 1$. We set $\alpha = 0.99$, as is most commonly done – this allows the reference distribution to closely model p while still including the desired smoothing property:

$$AlphaSkew(F_a, F_b) = \sum_{f \in F_b^+} [[w_b(f) \times log_e \frac{w_b(f)}{(1 - \alpha)w_b(f) + \alpha w_a(f)}] \tag{4.34}$$

As with KL-Divergence (Section 4.4.22), we also present the alternative version of calculating α-skew in the opposite direction:

$$AlphaSkewR(F_a, F_b) = \sum_{f \in F_a^+} [[w_a(f) \times log_e \frac{w_a(f)}{(1 - \alpha)w_a(f) + \alpha w_b(f)}] \tag{4.35}$$

4.4.25 Manhattan distance ◇

Manhattan distance, also known as the L1-norm, measures the element-wise absolute difference between two vectors:

$$Manhattan(F_a, F_b) = \sum_{f \in F_a \cup F_b} |w_a(f) - w_b(f)| \tag{4.36}$$

4.4.26 Euclidean distance ◇

Euclidean distance, also known as the L2-norm, is calculated as the length of the difference between the two input vectors:

$$Eculidean(F_a, F_b) = \sqrt{\sum_{f \in F_a \cup F_b} (w_a(f) - w_b(f))^2} \tag{4.37}$$

4.4.27 Chebyshev distance ◇

Chebyshev distance, also known as L_∞-norm, is a measure where the distance between two vectors is the maximum of their absolute differences along any coordinate dimension:

$$Chebyshev(F_a, F_b) = \max_{f \in F_a \cup F_b} (|w_a(f) - w_b(f)|) \tag{4.38}$$

4.5 Proposed measure: Weighted Cosine

Given a term pair $(a \rightarrow b)$, Kotlerman *et al.* (2010) refer to the set of features of the narrower term a as *tested features*, since they are tested for inclusion. Amongst these features, those found in the feature vector of the broader term b are denoted *included features*. They hypothesise that a directional distributional similarity should capture the following desirable properties:

1. The relevance of the included features to the narrower term.

2. The relevance of the included features to the broader term.

3. That inclusion detection is less reliable if the number of features of either the narrower or the broader term is small.

They show that existing measures which correspond to these criteria perform better and suggest a new directional similarity measure (BalAPInc) based on these principles. However, it is interesting to note that these properties do not explicitly mention any directional aspects of the measure, and symmetric similarity scores can also fulfil the requirements. We suggest two additions to this list of properties, one of which specifically targets the asymmetric properties of the desired similarity measures:

4. The included features are more important to the directional score calculation, compared to non-included features.

5. Highly weighted features of the broader term are more important to the score calculation, compared to features of the narrower term.

Most existing directional similarity scores measure how well the features of the narrower term are modelled by the broader term. If a entails b, then it is assumed that the possible contexts of a are also included by b, but b occurs in a wider range of contexts compared to a. This intuition is used by directional measures such as $ClarkeDE$, $WeedsPrec$, $APInc$ and $BalAPInc$.

In contrast, we found that many features for the narrower term are often highly specific to that term and do not generalise even to hypernyms. For example, the top three features for *cheese* in our dataset were *grana*, *feta* and *parmesan*. These features are unlikely to be found together with any other word or phrase, even with the hypernyms of *cheese*, such as *dairy product* and *food*. Since these features have a very high weight for the narrower term, their absence with the broader term will have a big negative impact on the similarity score.

We hypothesise that many terms have certain individual features that are common to them but not to other related words. Since most weighting schemes reward high relative co-occurrence, these features are also likely to receive high weights. Therefore, we suggest that features which are not found for both terms should have a decreased impact on the score calculation, as many of them are not expected to be shared between hyponyms and hypernyms. However, removing

them completely is also not advisable, as they allow the measure to estimate the overall relative importance of the included features to the specific term.

In addition, we believe that among the included features, those ranked higher for the broader term are more important to the directional measure. In the hyponymy relation ($a \rightarrow b$), the term b is more general and covers a wider range of semantic concepts. This also means it is more likely to be used in contexts that apply to different hyponyms of b. For example, some of the high-ranking features for *food* are *blandly-flavoured*, *high-calorie* and *uneaten*. These are properties that co-occur often with the term $food$, but can also be applied to most hyponyms of *food*, such as *cheese*, *meat*, *omelette* or *burger*. Therefore, we hypothesise that the presence of these features for the narrower term is a good indication of a hyponymy relation. This is somewhat in contrast to most previous work, where the weights of the narrower terms have been used as the main guideline for similarity calculation. However, our experiments with the new similarity measure, based on this hypothesis, manage to deliver improved performance on the hyponym generation task.

Cosine similarity is one of the symmetric similarity measures which corresponds to the first three desired properties. Our experiments also showed that it performs remarkably well at the tasks of hyponym detection and generation. Therefore, we decided to modify cosine similarity to also reflect the final two properties and produce an asymmetric similarity score. The standard feature vectors for each word contain weights indicating how important this feature is to the word. In addition, we can also specify weights for every feature that measure how important the feature is to that specific directional relation between the two terms. Weighted cosine similarity can then be used to calculate a modified similarity score:

$$WCos(F_a, F_b) = \frac{\sum_{f \in F_a \cap F_b} (z(f) \times w_a(f)) \times (z(f) \times w_b(f))}{\sqrt{\sum_{f \in F_a} (z(f) \times w_a(f))^2} \times \sqrt{\sum_{f \in F_b} (z(f) \times w_b(f))^2}} \quad (4.39)$$

where $z(f)$ is an additional weight for feature f, given the directional word pair (a, b).

Based on the new desired properties we want to downweight the importance of features that are not present for both terms. For this, we choose the simple solution of scaling them with a small constant $C \in [0, 1]$. Next, we also want to assign higher $z(f)$ values to the shared features that have high weights for the broader term b. We use the relative rank of feature f in F_b, $r_b(f)$, as the indicator of its importance and scale this value to the range from C to 1. This results in the importance function decreasing linearly as the rank number increases, but the weights for the included features always remain higher compared to the non-included features. Tied feature values are handled by assigning them the average rank value, following Spearman's rank correlation coefficient (Section 4.4.3). Adding 1 to the denominator of the relative rank calculation avoids exceptions with empty vectors, and also ensures that the value will always be strictly greater than C. The resulting function is as follows:

$$z(f) = \begin{cases} (1 - \frac{r_b(f)}{|F_b|+1}) \times (1 - C) + C & \text{if } f \in F_a \cap F_b \\ C & \text{otherwise} \end{cases} \tag{4.40}$$

The parameter C shows the relative importance of the 'unimportant' features to the directional relation. Setting it to 0 will ignore these features completely, while setting it to 1 will result in the traditional cosine measure. Experiments on the development data showed that the exact value of this parameter is not very important, as long as it is not too close to the extreme values of 0 or 1. We use the value $C = 0.5$ for reporting our results, meaning that the non-included features are half as important, compared to the included features.

4.6 Datasets

Zhitomirsky-Geffet & Dagan (2009) created a manually-annotated dataset for evaluating detection of lexical entailment. They randomly selected a set of 30 nouns and found 40 most similar words for each of them, according to two versions of the Lin measure. Every unique word pair was duplicated in both directions and manually annotated for lexical entailment. While the definition of entailment is broader than hyponymy, their end-goal of finding substitutable word pairs is very similar to ours. However, the Minipar (Lin, 1993) parser was used to construct the

dataset, generating some multi-word expressions which are not easily reproduced when finding distributional features with other parsers. We also report results on their dataset, but run our experiments on the subset of word pairs that contain only single terms (886 positive and 2,154 negative word pairs). We are thankful to Zhitomirsky-Geffet & Dagan for generously providing us with their dataset.

As WordNet (Miller, 1995) contains numerous manually annotated hyponymy relations, we can use it to construct suitable datasets for evaluating both hyponym detection and generation. While WordNet terms are annotated with only the closest hyponyms, we are considering all indirect/inherited hyponyms to be relevant – for example, given relations ($dog \rightarrow mammal$) and ($mammal \rightarrow animal$), then *dog* is also counted as a hyponym of *animal*. In addition, since synonymy can be thought of as a symmetric *is-a* relation, synonyms are counted as hyponyms in both directions.

For evaluating hyponym detection, we created a balanced dataset of positive and negative word pairs. As WordNet contains annotation for numerous words that rarely occur in practical datasets, we filtered out any terms that occurred less than 10 times in the British National Corpus (BNC), also considering the general part-of-speech tag (noun or verb) as tagged by RASP (Briscoe *et al.*, 2006). For every word we extract sets of positive and negative hyponym examples. The positive examples for a term consist of all its annotated synonyms and hyponyms. WordNet relations exist between synsets, but we refrain from the task of word sense disambiguation and count a candidate as a positive example if it is valid for any synset that contains the term.

There exists no explicit annotation for absence of hyponymy, and using random words would risk accidentally choosing correct examples that have not yet been annotated. However, we can make use of existing relations that directly contradict the possibility of hyponymy. As negative instances we use antonyms, together with hyponyms of hypernyms – for example, if ($mammal \rightarrow cat$) is added to the previous example, then *cat* can be used as a negative hyponymy instance for *dog*. Any terms that were present in the positive, as well as the negative set, were removed from both. We then randomly selected 10 positive and 10 negative examples for the final dataset, given that enough terms were available in both sets.

Zhitomirsky-Geffet & Dagan (2009) extracted only terms with the highest distributional similarity to annotate for the dataset. This somewhat changes the task, as all the candidates are distributionally very similar, and the goal is then to distinguish between the possible directions of the entailment relation. In contrast, we would like to evaluate how well similarity measures perform at finding hyponyms in general, not only the ones that have a high symmetric similarity. Our hyponym detection dataset does not make use of any initial similarity measure to filter terms, and allows us to evaluate the performance of each measure more directly. However, the dataset is still indirectly influenced by distributional similarity, as the methods for finding both positive and negative examples are likely to return terms that occur in similar contexts. Of course, this can be considered the hardest possible scenario for hyponym classification, but systems optimised only for this task can perform worse when applied to real data and practical applications.

To better model the scenario of hyponym generation, we create a second dataset suitable for a more open-ended task. The system will be given a single word as input and it will have to return all its hyponyms. This can be addressed as a classification task, returning the putatively correct set, or as a ranking task, returning a scored list of words with the correct hyponyms ranked highest. In contrast to the balanced detection dataset, all correct hyponyms are retained in the generation dataset, but all negative examples are discarded. We only included hypernyms that have at least 10 hyponyms, such that each of the hyponyms occurs at least 10 times in the BNC; this was done to limit the ratio of very rare words. The hypernym itself also needed to pass the frequency criteria. However, given that the hypernym matches the conditions, then the dataset contains all its direct and indirect hyponyms present in WordNet, including terms that were not found in the BNC.

While nearly all work on lexical entailment and hyponymy detection has been done on nouns, WordNet also includes annotation for verbs. Therefore, we created corresponding datasets for both nouns and verbs. In order to better facilitate future research and experiments, we randomly separated them into training (40%), development (30%), and test (30%) sets. The final size of all datasets is shown in Table 4.1. We make all the datasets available for download online,

along with descriptions of additional similarity measures, the complete table of results, the code used to perform the experiments, and the implementations of all the similarity measures (see Sections 2.2.1 and 2.2.2 for more information).[1]

	Train	Dev	Test
Noun detection	23120	17300	17300
Verb detection	10920	8160	8160
Noun generation	1230	922	922
Verb generation	594	444	444

Table 4.1: Dataset sizes for hyponym detection and generation. The detection datasets are measured in word pairs; the generation datasets are measured in the number of hypernyms, regardless of their hyponyms.

4.7 Experiments

4.7.1 Hyponym detection

In order to evaluate hyponym detection and generation, we first built distributional feature vectors for every word. The BNC was used as the background corpus and parsed by the RASP parser (Briscoe *et al.*, 2006). All the terms were lowercased; numbers were grouped and substituted by more generic tokens: currency (*30*), enumeration (*1st*), number (*one*), and other numerical (*14.57*). We also use the assigned part-of-speech tag, convert it to a more general form (noun, verb, etc.) (Petrov *et al.*, 2011), and append it to the lemma. This is done to provide a basic level of word-sense disambiguation, but still avoid the sparsity problems related to more specific tags.

As context features for a term, we use every incoming and outgoing dependency relation, together with the connected term. For example, given the dependency relation *(play_V, dobj, guitar_N)*, the tuple *(>dobj, guitar_N)* was extracted as a feature for *play_V*, and *(<dobj, play_V)* as a feature for *guitar_N*. We use only features that occur more than once in the whole dataset, and weight them using pointwise mutual information to construct feature vectors for every term.

[1] www.cl.cam.ac.uk/~mr472/hyponyms/

Features with negative weights were retained, as they proved to be beneficial for some similarity measures; the formulas in Section 4.4 specify which measures are defined using only positive features.

The first experiments evaluate how well the distributional similarity measures perform at the task of hyponym detection. Given a collection of term pairs, the system needs to assign higher scores to the pairs that show a correct hyponymy example. Following the evaluation by Kotlerman *et al.* (2010), we measure performance by sorting all the pairs in descending order of scores, and calculating average precision (AP) over the whole dataset. A system that ranks all correct pairs higher than any of the incorrect ones would receive an AP of 1.0. As the distributional similarity measures require no training or development, we report all results on both of these datasets for better comparison. In addition, *LexEnt* is the lexical entailment dataset created by Zhitomirsky-Geffet & Dagan (2009). Table 4.2 contains the results, with the best measure for each dataset marked in bold.

We can observe that Cosine and DiceGen2, both symmetric similarity measures, deliver relatively high performance even though the task is to detect asymmetric relationships. They do not use any notion of directionality, yet these measures perform well simply by assigning a higher score to pairs that are more distributionally similar. DiceGen2 also gives consistently better results on the verb datasets. BalAPInc and BalPrec are some of the best-performing directional measures, with BalAPInc achieving the highest performance on the Noun-Dev dataset.[1] Measures based on correlation (Spearman, Pearson, KendallsTau) do not appear to be well-suited for this task and give some of the lowest results on all datasets. Finally, the new WeightedCosine measure gives the highest performance on two different datasets, Noun-Train and LexEnt, and it is the second best-performing measure on both verb datasets.

Cosine, DiceGen2, BalAPInc and WeightedCosine were selected as the highest performing measures, and we also report their results on the designated test set in Table 4.3. The performances are similar to the previous results, with some

[1]Performance of BalAPInc on the lexical entailment dataset is not directly comparable to the results reported by Kotlerman *et al.* (2010). We only evaluate on single terms, whereas they include some multi-word expressions. In addition, our feature vectors are built from BNC parsed with the RASP parser, whereas they used the Reuters RCV1 corpus with Minipar.

	Noun		Verb		
	Train	**Dev**	**Train**	**Dev**	**LexEnt**
Cosine	58.76	57.23	57.50	57.33	47.56
Pearson	49.10	50.15	48.95	48.63	28.42
Spearman	48.74	49.70	48.80	48.48	27.93
JaccardSet	55.11	53.02	54.69	54.58	41.59
Lin	56.44	54.23	55.63	55.47	44.77
DiceSet	55.11	53.02	54.69	54.58	41.59
OverlapSet	50.84	51.77	50.52	50.25	31.76
CosineSet	56.03	54.13	55.49	55.44	43.30
JaccardGen	56.88	54.79	56.08	55.82	45.02
DiceGen	56.88	54.79	56.08	55.82	45.02
DiceGen2	58.89	56.93	**58.53**	**58.07**	46.66
KendallsTau	48.76	49.71	48.85	48.51	27.98
ClarkeDE ⋆	57.40	58.04	55.74	56.24	41.96
WeedsPrec ⋆	54.45	55.06	52.84	52.89	37.76
WeedsRec ⋆	48.74	48.00	49.77	49.49	29.52
WeedsF	56.25	54.03	55.50	55.34	44.05
AP ⋆	56.66	54.89	55.94	55.87	44.18
APInc ⋆	55.10	55.75	53.29	53.50	39.82
BalAPInc ⋆	58.80	**58.81**	56.83	56.82	45.04
LinD ⋆	55.11	55.68	53.37	53.49	38.96
BalPrec ⋆	58.49	57.89	57.16	56.96	45.61
KLDivergence ◇ ⋆	52.57	53.09	52.19	51.79	33.62
KLDivergenceR ◇ ⋆	47.99	47.27	48.66	48.79	28.78
JSDivergence ◇	52.89	53.21	51.73	51.94	25.19
AlphaSkew ◇ ⋆	49.72	49.76	49.89	49.97	23.21
AlphaSkewR ◇ ⋆	52.15	53.05	50.74	51.23	28.36
Manhattan ◇	52.88	53.19	51.71	51.92	25.13
Euclidean ◇	52.77	53.13	51.67	51.83	25.22
Chebyshev ◇	49.81	48.63	50.61	51.04	34.51
WeightedCosine ⋆	**59.09**	57.55	57.90	57.73	**47.62**

Table 4.2: Average precision of different similarity measures when performing hyponym detection, reported on the training, development, and lexical entailment datasets.

	Noun	Verb
Cosine	58.65	56.36
DiceGen2	58.60	**57.26**
BalAPInc ⋆	58.65	56.79
WeightedCosine ⋆	**59.08**	56.57

Table 4.3: Average precision of selected measures on the hyponym detection test datasets.

measures performing slightly better on nouns and others on verbs. WeightedCosine achieves the highest average precision on nouns (59.08%), while DiceGen2 performs best on verbs (57.26%).

4.7.2 Hyponym generation

Next, we evaluate how well these similarity measures perform at the task of hyponym generation. Given one input word at a time, the system needs to return a scored list of terms, such that the correct hyponyms of the original word are ranked higher. Average Precision is calculated for each returned list, and then averaged over all lists, resulting in Mean Average Precision (MAP). The MAP measure is typically used for evaluating the ranking of a fixed set, not in tasks where the set of returned items can vary. However, by penalising the system for any unseen correct hyponyms, we are able to report results that are comparable to other systems using different background corpora, filtering methods, and ranking measures.

The set of returned terms needs to be chosen by the system, and MAP assigns a higher penalty when a correct hyponym is not returned at all, compared to assigning it a low rank. We select as candidates all terms that have the correct part-of-speech (noun or verb), and occur at least 10 times in the corpus. This results in 45,986 terms for nouns and 6,011 terms for verbs; the input term itself is removed from the list before evaluation. The similarity measures are then used to assign scores to each candidate term, indicating the confidence that it is a valid hyponym of the input word. Table 4.4 shows results on the training and development sets.

The results show that WeightedCosine outperforms all other measures on

	Noun		Verb	
	Train	**Dev**	**Train**	**Dev**
Cosine	2.63	2.62	4.75	5.15
Pearson	0.13	0.14	0.71	0.69
Spearman	0.12	0.14	0.70	0.67
JaccardSet	1.37	1.47	3.01	3.02
Lin	1.99	2.03	3.81	3.95
DiceSet	1.37	1.47	3.01	3.02
OverlapSet	0.18	0.19	0.98	0.97
CosineSet	1.61	1.64	3.60	3.61
JaccardGen	2.01	2.05	3.77	3.93
DiceGen	2.01	2.05	3.77	3.93
DiceGen2	2.48	2.50	4.45	4.64
KendallsTau	0.12	0.14	0.70	0.67
ClarkeDE ⋆	0.24	0.26	1.14	1.14
WeedsPrec ⋆	0.15	0.17	0.67	0.68
WeedsRec ⋆	0.79	0.80	2.25	2.08
WeedsF	1.87	1.93	3.61	3.76
AP ⋆	1.66	1.67	3.71	3.70
APInc ⋆	0.18	0.20	0.79	0.78
BalAPInc ⋆	1.50	1.51	3.08	2.84
LinD ⋆	0.16	0.18	0.72	0.73
BalPrec ⋆	1.73	1.74	3.29	2.99
KLDivergence ◇ ⋆	0.56	0.60	1.43	1.31
KLDivergenceR ◇ ⋆	0.39	0.38	1.48	1.38
JSDivergence ◇	0.13	0.16	0.52	0.53
AlphaSkew ◇ ⋆	0.81	0.83	2.30	2.11
AlphaSkewR ◇ ⋆	0.08	0.10	0.46	0.46
Manhattan ◇	0.09	0.11	0.47	0.47
Euclidean ◇	0.09	0.11	0.47	0.47
Chebyshev ◇	0.23	0.27	1.34	1.27
WeightedCosine ⋆	**2.73**	**2.72**	**4.80**	**5.18**

Table 4.4: Mean average precision of different similarity measures when performing hyponym generation, reported on the training and development datasets.

all datasets. Cosine and DiceGen2 also give high results, whereas the other measures have considerably lower MAP scores. BalPrec and AP are some of the highest among directional similarity measures, but are still outperformed by several symmetric measures. It is interesting to note that AP consistently outperforms APInc and BalAPInc, which are designed to be improvements of the original measure. This is likely caused by APInc assigning high similarity scores to pairs where the narrower term has a much lower frequency compared to the broader term. Measures such as ClarkeDE and WeedsPrec are also vulnerable to the same problem, as they quantify the total similarity relative to the sum of feature weights for the narrower term.

In Table 4.5 we present results of the same high-performing measures on the test sets. WeightedCosine achieves the highest performance on the nouns (2.88%), and shares the highest rank with Cosine on the verbs (5.52%).

	Noun	**Verb**
Cosine	2.80	**5.52**
DiceGen2	2.63	4.83
BalAPInc	1.58	2.84
WeightedCosine	**2.88**	**5.52**

Table 4.5: Mean average precision of selected measures on the test datasets.

We measure the statistical significance of the change in MAP by using the Approximate Randomisation Test (Noreen, 1989; Cohen, 1995) with 10^6 iterations. This method is a computer-intensive statistical hypothesis test, and is the recommended significance test for measuring MAP differences in the Information Retrieval task (Smucker *et al.*, 2007). It is designed to assess result differences with respect to a test statistic in cases where the sampling distribution of the test statistic is unknown. The performance of WeightedCosine was found to be significantly better ($p < 0.05$) on all noun datasets (train, dev and test) in the generation task, when compared to the second ranking measure (Cosine). The differences on the verb datasets were not significant.

The numerical values of the MAP scores are noticeably low for all measures on the hyponym generation task due to a number of reasons. First, the task is rather difficult for an unsupervised similarity measure, and the MAP score is calculated

over a very large number of candidates (45,985 in the case of nouns). Second, as WordNet is a manually annotated resource, there are presumably many correct hyponym relations that have not been added yet, and the system will always be penalised for giving these terms a high score. In addition, there are many words present in WordNet that do not occur in our corpus, and are therefore not even considered as potential hyponyms. Finally, the similarity measures assign scores based on how well one term can be substituted for another, by comparing their context vectors, and these terms might not always be strictly hyponyms. However, we assume that distributional similarity measures which assign valid scores to WordNet-annotated hyponym relations, are more likely to also provide better scores to other term pairs that may be lexically substitutable. By normalising MAP over the total number of correct hyponyms in the dataset, we are able to report results that are comparable even to very different systems using alternative background corpora and filtering methods. While ontology creation systems are commonly evaluated only using precision, this approach also takes recall and ranking into account.

Table 4.6 contains some examples of hyponym generation, ordered by their score, illustrating the properties of each similarity measure. A larger set of examples can be found in Appendix B. Cosine and DiceGen2 return both wide (*food*) and more specific concepts (*garlic*), whereas BalAPInc clearly prefers narrow and rare terms. WeightedCosine closely follows Cosine, but also includes a more subtle preference towards narrow concepts.

4.7.3 Supervised learning

Previous sections investigated hyponym detection and generation using distributional similarity measures. All of these measures use only a large unannotated corpus for collecting distributional features, and require no labelled data. In order to see how these unsupervised methods compare to fully-supervised learning, we also constructed experiments with Support Vector Machines (SVM) (Vapnik, 1982; Cortes & Vapnik, 1995).

We utilise SVM^{Light} (Joachims, 1999) and train separate models for hyponym detection and generation, using a linear kernel and the designated training set.

	vegetable (noun)	**travel (verb)**
Cosine	fruit, salad, **potato**, food, **lettuce**, herb, ingredient, meal, meat, dish	**move**, **walk**, **go**, **work**, **pass**, **run**, **return**, live, **drive**, **fly**
DiceGen2	salad, fruit, **potato**, **lettuce**, herb, ingredient, meat, **mushroom**, **tomato**, garlic	**walk**, **fly**, arrive, **move**, attend, **drive**, **ride**, stay, **pass**, **cross**
BalAPInc ⋆	**veg**, caulerpa, worm-cake, cryptocorynes, roughage, parmesan, caviare, lectin, crudit, owenites	**journey**, **trudge**, **stray**, **tramp**, queue, urinate, **stroll**, **wander**, **taxi**, cooperate
WeightedCosine ⋆	fruit, salad, **lettuce**, **potato**, food, herb, prawn, meat, ingredient, meal	**move**, **walk**, **pass**, **work**, **drive**, **fly**, **go**, **return**, live, **run**

Table 4.6: Examples of hyponym generation for the noun *vegetable* and the verb *travel*. Correct hyponyms, according to WordNet, are marked in bold.

Each item being classified corresponds to one directional word pair. As features for every word pair, we used all the similarity measures described in the previous sections. This method is based on Berant *et al.* (2010) who also calculate a wide selection of different similarity measures and use them as features for an SVM classifier. In addition, we included the following features for both individual words and multiplied for the paired words:

- Frequency
- Number of features
- Number of features relative to frequency
- Number of shared features relative to number of total features

All feature values were normalised by subtracting the mean and dividing by the range. As the generation dataset does not contain any negative examples, we created a balanced dataset for training by randomly sampling words that were not annotated as hyponyms. We also experimented with alternative methods for choosing negative examples, based on the selection method for hyponym detection, but found random sampling to give the best performance. The results are presented in Table 4.7.

	Noun			Verb		
	Train	**Dev**	**Test**	**Train**	**Dev**	**Test**
Detection	63.73	63.46	64.00	64.44	63.03	62.68
Generation	1.88	1.89	1.97	4.64	4.78	4.97

Table 4.7: Experiments on hyponym detection and generation using Support Vector Machines.

The experiments show that a supervised learning method gives the best results for the hyponym detection task, achieving 64.00% on the noun-test dataset compared to 59.08% by WeightedCosine. However, it does not perform as well on the generation task, giving only 1.97% compared to 2.88% by WeightedCosine. Running generation experiments with SVMs is also rather time-consuming, as the system needs to classify all possible word pairs, therefore we find distributional similarity measures to be a better choice for the task of hyponym generation.

Finally, we also experimented with switching the models, e.g. training a model on the detection data and testing on generation, but this delivered lower results for both cases.

4.8 Conclusion

Given only a single term as input, hyponym generation is the task of finding correct hyponyms for that term. It has practical applications for a wide variety of NLP tasks. For example, information retrieval and information extraction can directly benefit from being able to find lexical substitutes and construct more specific versions of a query, while retaining the original semantics. Ontology creation, summarisation, question answering, entailment detection and language modelling can also use hyponyms to increase both accuracy and coverage.

In order to investigate the performance of various distributional similarity measures on the tasks of hyponym detection and generation, we created four new datasets using WordNet. They cover hyponym relations between both nouns and verbs, and are split into training, development and test sets. The hyponym detection datasets contain a balanced selection of correct and incorrect hyponymous word pairs. The hyponym generation datasets contain selected terms together

with all their hyponyms, leaving it up to the system to find the correct hyponym instances using unrestricted knowledge sources.

Hyponym generation can be performed by applying hyponym detection on a large number of candidate pairs. We evaluated and compared the performance of different distributional similarity measures at both hyponym detection and generation. Cosine and DiceGen2 similarity measures were found to perform remarkably well, considering that they are symmetric measures and the task assumes modelling of a directional relation. This implies that existing directional similarity measures are not able to capture hyponymy very well, and there is plenty of room for improvement. The directional measures of AP, BalAPInc and BalPrec performed well on the hyponym detection task, but were outranked by several other symmetric measures when doing hyponym generation.

We also proposed a novel similarity measure, based on cosine similarity, by applying additional directional weights to each feature. We hypothesised that the feature weights of the broader term are more indicative of the hyponymy relation, compared to the narrower term. In addition, we assigned lower importance to features that are not shared between the two terms, since many high-weighted features are often too specific to contribute to hyponymy detection. Our measure delivers significantly better results when doing hyponym generation on nouns, and gives comparable performance on verbs.

Finally, in order to compare these methods to a fully-supervised system, we trained an SVM classifier for solving the same tasks. All of the unsupervised similarity measures were included as features in the supervised model, and this configuration gave the highest performance for hyponym detection. However, the classifier was outperformed by the best individual similarity measures on hyponym generation. This result demonstrates that a theoretically-motivated approach without any supervision or annotated data can, in some cases, outperform a fully supervised method.

There is much room for additional improvement in both hyponym detection and generation. In addition to potential new similarity measures, the problems could also be approached through different methods, such as hyponym acquisition or more focussed supervised learning algorithms. It is also worth exploring whether nouns and verbs could benefit from different configurations, as they have

somewhat different distributions of context features. The task of hyponym generation is not yet sufficiently researched, and many existing NLP applications can directly benefit from more accurate solutions.

Chapter 5

Entailment Detection

5.1 Introduction

Understanding that two different texts are semantically similar has benefits for nearly all NLP tasks, including information retrieval, information extraction, question answering and summarisation. Similarity detection is usually performed either on single words (synonymy) or full sentences and paragraphs (paraphrasing). A symmetric similarity relation implies that both elements can be interchanged (synonymy and paraphrasing), while directional similarity suggests that one fragment can be substituted for the other but not the opposite (hyponymy and entailment).

All of these language phenomena can be expressed using a single entailment relation. For paraphrases and synonyms the relation holds in both directions (*observe* $\leftrightarrow$ *notice*), whereas entailment and hyponymy are modelled as a unidirectional relation (*overexpress* $\rightarrow$ *express*). In addition, such relations can be defined between text fragments of any size and shape, ranging from a single word to a complete text segment. For example (*argues against* $\rightarrow$ *does not support*), and (*the protein has been implicated* $\leftrightarrow$ *the protein has been shown to be involved*).

In this chapter we propose a new task – detecting entailment relations between any kinds of text fragments. A unified approach is not expected to perform better when compared to systems optimised only for a specific task (e.g., recognising entailment between sentences), but constructing a common theory to cover all text

fragments has important benefits. A broader approach will allow for entailment discovery among a much wider range of fragment types for which no specialised systems currently exist. In addition, entailment relations can be found between different types of fragments (e.g., a predicate entailing an adjunct). Finally, a common system is much easier to develop and integrate with potential applications compared to having a separate system for each type of fragment.

In Chapter 4 we investigated minimally supervised methods for detecting hyponym relations between single words. The current chapter extends this task to include larger dependency graph fragments, and we present a unified framework that can be used to detect entailment relations between sentence fragments of various types and sizes.

The final system is designed to work with anything that can be represented as a connected subgraph of dependency relations, including single words, constituents of various sizes, text adjuncts, predicates, relations and full sentences. The approach is completely unsupervised and requires only a large plain text corpus to gather statistics for calculating distributional similarity. This makes it ideal for the biomedical domain where the availability of annotated training data is limited. We apply these methods by using a large corpus of biomedical papers for feature collection and evaluate on a manually constructed dataset of entailing fragment pairs, extracted from biomedical texts.

5.2 Research goals

In order to analyse the task of entailment detection, and to build a working system, we have set the following research goals:

- Determine whether dependency graphs are suitable for extracting meaningful connected structures for entailment detection.
- Investigate whether the distributional hypothesis can be extended from single words to larger text fragments and, more specifically, to larger dependency graph fragments.

- Examine how different types of similarity measures perform with larger connected subgraphs, compared to single words.
- Find methods of integrating both distributional and lexical similarity when calculating the similarity between graph fragments.
- Determine whether the detection of sentence-level negation and hedging has a practical impact to the task of entailment detection.
- Develop and evaluate a unified minimally supervised approach for entailment detection.

5.3 Background

The entailment relation between two sentences or paragraphs is commonly defined following the guidelines in the Recognizing Textual Entailment (RTE) Challenge (Bar-Haim *et al.*, 2006):

(18) Text T entails hypothesis H if, typically, a human reading T would infer that H is most likely true.

For example, the following text (T) entails the hypothesis (H):

(19) T: Francis Crick, James D. Watson and Maurice Wilkins were awarded the Nobel Prize in 1962.
H: Francis Crick received an award.

When there is no valid entailment relation between sentences, it is due to one of two reasons: either there is not enough evidence to establish the relation, or one text contradicts the other. H1 and H2 in the example below illustrate both of these cases, respectively.

(20) T: Glutamine is the most abundant free amino acid in human blood.
H1: Glutamine is one of the most important sources of energy in cells.
H2: Glutamine is not found in human blood.

Over time, additional guidelines have been specified for the entailment detection challenge:

- Entailment is a directional relation. The hypothesis must be entailed from the text but the text does not need to be entailed from the hypothesis.
- The hypothesis must be fully entailed by the text and must not contain parts that cannot be entailed.
- Entailment allows presupposition of common world knowledge.
- If the inference is very probable but not certain, the hypothesis is still judged as being entailed from the text.
- Given the text and hypotheses might originate from documents at different points in time, tense aspects should be ignored.

This definition is somewhat subjective, as it assumes certain background knowledge and a shared understanding of language. However, human judges have shown very good agreement on the task of entailment recognition – 3 judges reached 91-96% agreement when independently annotating the dataset of RTE-1 (Dagan *et al.*, 2006). This indicates that humans have a good natural understanding of the concept of entailment, and automated systems are not yet close to that level.

Most existing work on entailment detection has focused on comparing full sentences or larger text units, and various methods have been developed to calculate their semantic similarity. For example, Haghighi *et al.* (2005) represent sentences as directed dependency graphs and use graph matching to measure semantic overlap. Their method also compares the dependency relations present in both sentences, thereby incorporating extra syntactic information in the similarity calculation. Hickl *et al.* (2006) combine lexico-syntactic features and automatically acquired paraphrases to classify entailing sentences. Lintean & Rus (2009) make use of weighted dependencies and word semantics to detect paraphrases. In addition to similarity, they look at dissimilarity between two sentences and use their ratio as the confidence score for paraphrasing.

To a lesser extent, the detection and extraction of entailment examples between relations has also been investigated. Lin & Pantel (2001) were one of the first to extend the distributional hypothesis to dependency paths, and they construct an unsupervised algorithm for Discovering Inference Rules from Text (often referred to as the DIRT algorithm). Szpektor *et al.* (2004) further extend this work and describe the TE/ASE method for extracting entailing relation templates from the Web. The distributional similarity of argument terms has also been used for other related tasks, such as detecting entailment between unary templates (Szpektor & Dagan, 2008), and to learn a transitive entailment graph of binary relations (Berant *et al.*, 2010).

Finally, there has also been some work on detecting directional similarity relations between single words. Most methods involve syntactic pattern matching to automatically discover word-level hypernym pairs (e.g., Snow *et al.*, 2005). More recently, distributional similarity methods have also been used to detect directional inference relations between single words (Kotlerman *et al.*, 2010). We explored the tasks of hyponym detection and generation in more detail in Chapter 4, and additional related research is presented in Section 4.3.

In contrast to our approach, each of the approaches described above only focuses on detecting entailment between specific subtypes of fragments (either sentences, relations or words) and optimising the system for a single scenario. This means only limited types of entailment relations are found and they cannot be used for entailment generation or compositional entailment detection, as described in Section 5.4.

MacCartney & Manning (2008) approach sentence-level entailment detection by breaking the problem into a sequence of atomic edits linking the premise to the hypothesis. Entailment relations are then predicted for each edit, propagated up through a syntax tree and then used to compose the resulting entailment decision. However, their system focuses more on natural logic and uses a predefined set of compositional rules to capture a subset of valid inferences with high precision but low recall. It also relies on a supervised classifier and information from WordNet to reach the final entailment decision.

Androutsopoulos & Malakasiotis (2010) have assembled a survey of different tasks and approaches related to paraphrasing and entailment. They describe three

different goals (paraphrase recognition, generation and extraction) and analyse various methods for solving them.

5.4 Applications

Entailment detection between fragments is a vital step towards entailment generation – given text T, the system will have to generate all texts that either entail T or are entailed by T. This is motivated by applications in information retrieval (IR) and information extraction (IE). For example, if we wish to find all genes that are synthesised in the larval tissue, the following IE query can be constructed (with x and y marking unknown variables):

(21) x is synthesised in the larval tissue

Known entailment relations can be used to modify the query:

- *overexpression* $\rightarrow$ *synthesis*
- *larval fat body* $\rightarrow$ *larval tissue*
- *the synthesis of* x *in* $y \leftrightarrow$ x *is synthesised in* y

Pattern (22) entails pattern (21) and would also return results matching the information need.

(22) the overexpression of x in the larval fat body

A system for entailment detection can automatically extract a database of entailing fragments from a large corpus and use them to modify any query given by the user. Recent studies have also investigated how complex sentence-level entailment relations can be broken down into smaller consecutive steps involving fragment-level entailment (Sammons *et al.*, 2010; Bentivogli *et al.*, 2010). For example:

(23) T: The mitogenic effects of the B beta chain of fibrinogen are mediated through cell surface calreticulin.
H: Fibrinogen beta chain interacts with CRP55.

To recognise that the hypothesis is entailed by the text, it can be decomposed into five separate steps involving text fragments:

1. *B beta chain of fibrinogen* → *Fibrinogen beta chain*
2. *calreticulin* → *CRP55*
3. *the mitogenic effects of x are mediated through y* → *y mediates the mitogenic effects of x*
4. *y mediates the mitogenic effects of x* → *y interacts with x*
5. *y interacts with x* → *x interacts with y*

This illustrates how entailment detection between various smaller fragments can be used to construct an entailment decision between more complicated sentences. However, only the presence of these constructions has been investigated and, to the best of our knowledge, no models currently exist for automated detection and composition of such entailment relations.

5.5 Modelling entailment between graph fragments

In order to cover a wide variety of language phenomena, a fragment is defined as follows:

Definition 1 *A fragment is any connected subgraph of a directed dependency graph containing one or more words and the grammatical relations between them.*

This definition is intended to allow extraction of a wide variety of fragments from a dependency tree or graph representation of sentences found using any appropriate parser capable of returning such output (e.g., Kübler *et al.*, 2009; Briscoe *et al.*, 2006; Curran *et al.*, 2007; Nivre *et al.*, 2007). The definition covers single- or multi-word constituents functioning as dependents (e.g., *sites*, *putative binding sites*), text adjuncts (*in the cell wall*), single- or multi-word predicates (* *binds to receptors in the airways*) and relations (* *binds and activates* *),

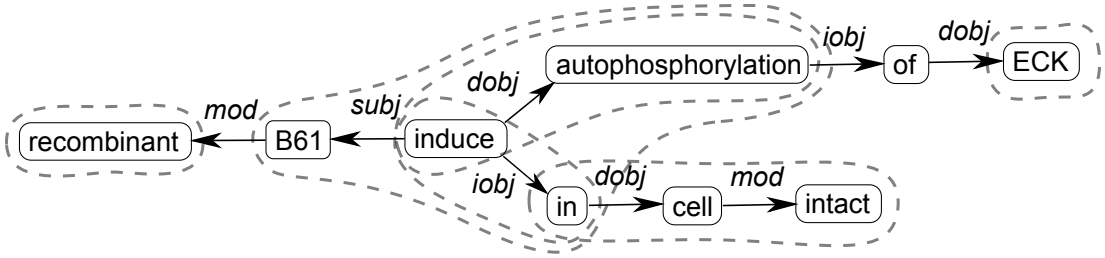

Figure 5.1: Dependency graph for the sentence: *Recombinant B61 induces autophosphorylation of ECK in intact cells.* Some interesting fragments are marked by dotted lines.

including ones with 'internal' dependent slots (** inhibits * at **), some of which may be fixed in the fragment (** induces autophosphorylation of * in * cells*), and also full sentences.[1]

Dependency graphs are naturally well-suited for the extraction of smaller subfragments, as can be seen from the example in Figure 5.1. The structure is centred at the main verb, and each outgoing edge adds more specific information, usually in the order of decreasing importance (modifiers are positioned further compared to the main verbs and nouns). Splitting any of the dependency relations automatically creates two smaller structures that are still semantically coherent, even though the textual representation may be broken. For example, removing the central *dobj* relation creates two fragments: a noun phrase (*autophosphorylation of ECK*), and a verb phrase with an unspecified object (*Recombinant B61 induces * in intact cells*). This property also applies when futher separating the fragments recursively. Similar splits of the textual representation would easily create numerous incoherent sequences, such as *'of ECK in intact'*.

Our aim is to detect semantically similar fragments which can be substituted for each other in text, resulting in more general or more specific versions of the same proposition. This kind of similarity can be thought of as an entailment relation and we define entailment between two fragments as follows:

[1]The asterisks (*) are used to indicate missing dependents in order to increase the readability of the fragment when represented textually. The actual fragments are kept in graph form and have no need for them.

Definition 2 *Fragment A entails fragment B ($A \rightarrow B$) if A can be replaced by B in a sentence S and the resulting sentence S' can be entailed from the original one ($S \rightarrow S'$).*

This also requires estimating entailment relations between sentences, for which we use the definition established by Bar-Haim *et al.* (2006):

Definition 3 *Text T entails hypothesis H ($T \rightarrow H$) if, typically, a human reading T would infer that H is most likely true.*

It is important to note that the sentence S must not contain hedging, negation, quantifiers, or other constructions that can modify the properties of the entailment relation.

We model the semantic similarity of fragments as a combination of two separate directional similarity scores:

1. **Intrinsic similarity**: how similar are the components of the fragments.
2. **Extrinsic similarity**: how similar are the contexts of the fragments.

To find the overall score, these two similarity measures are combined linearly using a weighting parameter α:

$$Score(A \rightarrow B) = \alpha \times IntSim(A \rightarrow B) + (1 - \alpha) \times ExtSim(A \rightarrow B)$$

We use the notation $f(A \rightarrow B)$ to indicate an asymmetric function between A and B. When referring only to single words, lowercase letters (a,b) are used; when referring to fragments of any size, including single words, then uppercase letters are used (A,B).

$Score(A \rightarrow B)$ is the confidence score that fragment A entails fragment B – higher score indicates higher confidence and 0 means no entailment. $IntSim(A \rightarrow B)$ is the intrinsic similarity between two fragments. It can be any function that compares them internally, for example by matching the structure of one fragment to another, and outputs a similarity score in the range of $[0, 1]$. $ExtSim(A \rightarrow B)$ is a measure of extrinsic similarity that compares the contexts of the two

fragments. α is set to 0.5 for an unsupervised approach, but the effects of tuning this parameter are further analysed in Section 5.7.

The directional similarity score is first found between words in each fragment, which are then used to calculate the overall score between the two fragments.

5.5.1 Intrinsic similarity

$IntSim(A \rightarrow B)$ is defined as the intrinsic similarity between the two words or fragments. In order to best capture entailment, this measure should be asymmetrical. We use the following simple formula for word-level score calculation:

$$IntSim(a \rightarrow b) = \frac{length(c)}{length(b)}$$

where c is the longest common substring for a and b. This measure will show the ratio of b that is also contained in a. For example:

$$IntSim(overexpress \rightarrow expression) = 0.70$$

$$IntSim(expression \rightarrow overexpress) = 0.64$$

We found that measuring the length of the longest shared substring worked better, compared to the more widely used Levenshtein distance measure (Levenshtein, 1966), also known as string edit distance. For example, Levenshtein distance will assign a high similarity to the word pair (*overexpress*, *overcompress*), even though they have different lexical stems and very different meaning. In contrast, our IntSim measure will return a relatively low similarity, as it matches only on the common suffix.

The intrinsic similarity function for fragments is defined using an injective function between components of A and components of B:

$$IntSim(A \rightarrow B) = \frac{Mapping(A \rightarrow B)}{|B|}$$

where $Mapping(A \rightarrow B)$ is a function that goes through all the possible word pairs $\{(a, b)|a \in A, b \in B\}$ and at each iteration connects the one with the

```
while unused elements in A and B do
  bestScore = 0
  for a ∈ A, b ∈ B, a and b are unused do
    if Score(a → b) > bestScore then
      bestScore = Score(a → b)
    end if
  end for
  total+ = bestScore
end while
return total
```

Figure 5.2: Pseudocode for mapping between two fragments.

highest entailment score, returning the sum of those scores. Figure 5.2 contains pseudocode for the mapping process. Dividing the value of $Mapping(A \rightarrow B)$ by the number of components in B gives an asymmetric score that indicates how well B is covered by A. It returns a lower score if B contains more elements than A, as some words cannot be matched to anything. While there are exceptions, it is common that if B is larger than A, then it cannot be entailed by A as it contains more information.

The word-level entailment score $Score(a \rightarrow b)$ is directly used to estimate the entailment score between fragments, $Score(A \rightarrow B)$. In this case we are working with two levels – fragments which in turn consist of words. However, this can be extended to a truly recursive method where fragments consist of smaller fragments.

5.5.2 Extrinsic similarity

The extrinsic similarity between two fragments or words is modelled using measures of directional distributional similarity. We define a context relation as a tuple (a, d, r, a') where a is the main word, a' is a word connected to it through a dependency relation, r is the label of that relation and d shows the direction of the relation. The tuple $f : (d, r, a')$ is referred to as a feature of a.

To calculate the distributional similarity between two fragments, we adopt an approach similar to Weeds *et al.* (2005). Using the previous notation, (d, r, a') is a feature of fragment A if (d, r, a') is a feature for a word which is contained in

A. The general algorithm for feature collection is as follows:

1. Find the next instance of a fragment in the background corpus.
2. For each word in the fragment, find dependency relations which connect to words not contained in the fragment.
3. Count these dependency relations as distributional features for the fragment.

For example, in Figure 5.1 the fragment *(* induces * in *)* has three features: *(1, subj, B61)*, *(1, dobj, autophosphorylation)* and *(1, dobj, cell)*. The value 1 indicates an outward-directed relation.

The BioMed Central corpus of full papers was used to collect distributional similarity features for each fragment. 1,000 papers were randomly selected and separated for constructing the test set, leaving 70,821 biomedical full papers. These were tokenised and parsed using the RASP system (Briscoe *et al.*, 2006) in order to extract dependency relations.

We experimented with different schemes for feature weighting and found the best one for this task to be a variation of Dice's coefficient (Dice, 1945), described by Curran (2003):

$$w_A(f) = \frac{2P(A, f)}{P(A, *) + P(*, f)} \tag{5.1}$$

where $w_A(f)$ is the weight of feature f for fragment A, $P(*, f)$ is the probability of the feature appearing in the corpus with any fragment, $P(A, *)$ is the probability of the fragment appearing with any feature, and $P(A, f)$ is the probability of the fragment and the feature appearing together.

We evaluated 15 best-performing similarity measures from Chapter 4 for modelling extrinsic similarity, and found that words and fragments benefitted from different types of measures. $ClarkeDE$ (Clarke, 2009), described in Section 4.4.16, was used for fragment-level $ExtSim$ in the final system as it achieved the highest performance:

$$ClarkeDE(A \rightarrow B) = \frac{\sum_{f \in F_A \cap F_B} min(w_A(f), w_B(f))}{\sum_{f \in F_A} w_A(f)} \tag{5.2}$$

where F_A is the set of features for fragment A and $w_A(f)$ is the weight of feature f for fragment A. It quantifies the weighted coverage of the features of A by the features of B by finding the sum of minimum weights.

The $ClarkeDE$ similarity measure is designed to detect whether the features of A are a proper subset of the features of B. This works well for finding more general versions of relatively sparse fragments, but not when comparing words or fragments which are roughly equal paraphrases. As a solution we constructed a new measure based on the symmetrical Lin measure (Lin, 1998).

$$LinD(A \rightarrow B) = \frac{\sum_{f \in F_A \cap F_B} [w_A(f) + w_B(f)]}{\sum_{f \in F_A} w_A(f) + \sum_{f \in F_A \cap F_B} w_B(f)} \tag{5.3}$$

In the $LinD$ measure, described in Section 4.4.20, the features of B which are not found in A are excluded from the score calculation, making the score non-symmetrical but more balanced compared to $ClarkeDE$. We applied this for word-level distributional similarity and achieved better results compared to other common similarity measures.

Finally, we also use the $LinD$ similarity to better detect possible paraphrases between fragments. If this similarity is very high (greater than 85%), then a symmetric relationship between the fragments is assumed and the value of $LinD$ is used as $ExtSim$. Otherwise, the system reverts to the $ClarkeDE$ measure for handling unidirectional relations.

The system does not explicitly restrict two entailing fragments to have the same syntactic type or internal structures. However, the incorporation of the distributional similarity score will lead the system towards fragments that occur in similar contexts and have similar valency.

5.5.3 Hedging and negation

In general, adding more information to a text will produce a new proposition that still entails the original hypothesis. However, as Haghighi *et al.* (2005) point out, there are some important exceptions. Speculative language, also known as hedging, is a way of weakening the strength of a statement and can be one source of such exceptions.

In the following example, we have only added a modifier to the original term, but this is enough to invert the direction of the entailment relation:

(24) is repressed by → is affected by
may be repressed by ← is affected by

The same applies to negation cues, which are also capable of reversing the relation:

(25) biological discovery → discovery
no biological discovery ← no discovery

Such cases are handled in our system by inverting the direction of the score calculation if a fragment is found to contain a special cue word that commonly indicates hedged language or negation. In order to find the list of indicative hedge cues, we analysed the training corpus of CoNLL 2010 Shared Task (Farkas *et al.*, 2010), also used in Chapter 3, which is annotated for speculation cues and scopes. Any cues that occurred less than 5 times or occurred more often as normal text than as cue words were filtered out, resulting in the following list:

(26) *suggest, may, might, indicate that, appear, likely, could, possible, whether, would, think, seem, probably, assume, putative, unclear, propose, imply, possibly*

For negation cues we used the list collected by Morante (2009):

(27) *absence, absent, cannot, could not, either, except, exclude, fail, failure, favor over, impossible, instead of, lack, loss, miss, negative, neither, nor, never, no, no longer, none, not, rather than, rule out, unable, with the exception of, without*

This is a fast and basic method for estimating the presence of hedging and negation in a fragment. In Chapter 3 we described a more sophisticated system which could deliver more accurate results, especially when extended to negation scopes. However, in this experiment we wish to test whether the detection of hedging and negation in general will have any direct practical effect on entailment

detection. In addition, the exact scope detection is most beneficial for longer texts, whereas the presence of a keyword acts as a good indication of hedging and negation for relatively short fragments. Therefore, we leave the integration of these systems for future work, and present here the preliminary experiments with a simplified approach.

5.6 Dataset

We created a "pilot" dataset for evaluating different entailment detection methods between fragments, and it is publically available for download.[1] In order to look for valid entailment examples, 1,000 biomedical papers from the BioMed Central full-text corpus were randomly chosen and analysed. We hypothesised that two very similar sentences extracted from the same paper are likely to be more and less general versions of the same proposition. For example, the author may write the same statement in three separate locations: the introduction, the main article body, and the conclusion. Some of these statements tend to be more specific than others, or modified paraphrases of each other, providing a natural source of fragment-level entailment examples. To find these instances, we first use a bag-of-words approach to calculate the similarities between all sentences in a single paper. Next, ten of the most similar but non-identical sentence pairs from every paper were presented for manual review and 150 fragment pairs were created based on these sentences, 100 of which were selected for the final set. Negative instances were found using the same process, selecting fragment pairs in similar contexts that do not entail each other, but some fragments were slightly modified to make them substitutable in the same sentence.

The Microsoft Research (MSR) Paraphrase Corpus (Dolan & Brockett, 2005) is a similar dataset containing a collection of sentence pairs that have been manually annotated for paraphrases. The corpus was created by automatically extracting sentences based on high word similarity, and then suggesting them for manual labelling. It was found that on average 70% of the words in these sentence pairs are identical (Weeds *et al.*, 2005). Such a dataset will not contain a wide range of paraphrase types, since only sentences with high lexical overlap can pass

[1] www.cl.cam.ac.uk/~mr472/entailment/

```
1: These findings suggest that spinal AMPARs might participate in
   the central spinal mechanism of persistent inflammatory pain.
2:
3: might participate in
4: <-
5: are involved in
```

Figure 5.3: Example sentence pair from the dataset

the initial automatic filter. We employ a similar method to construct our dataset, but as our work involves smaller graph fragments compared to full sentences, the overlap problem is somewhat reduced. While all words in the sentence are used to perform the initial matching, most of them are discarded and only the non-identical fragments are retained. Experiments in Section 5.7 demonstrate that a simple bag-of-words approach performs rather poorly on the task, confirming that the extraction method produces a diverse selection of fragments.

Two annotators assigned one of four relation types to every candidate pair based on how well one fragment can be substituted for the other in text while preserving meaning ($A \leftrightarrow B$, $A \rightarrow B$, $A \leftarrow B$ or $A \neq B$). Cohen's Kappa between the annotators was 0.88, indicating very high agreement. Instances with disagreement were then reviewed and replaced for the final dataset.

Each fragment pair has two binary entailment decisions (one in either direction) and the set is evenly balanced, containing 100 entailment and 100 non-entailment relations. An example sentence with the first fragment is also included in the dataset. Fragment sizes range from 1 to 20 words, with the average of 2.86 words.

Figure 5.3 shows one example from the dataset. The first line contains the sample sentence, followed by the first fragment, the direction of the relation, and the second fragment. The fragments are not stored in their final graph form in order to enable the use of different dependency parsers, but they are chosen to produce connected dependency graphs.

5.7 Experiments

The system assigns a score to each entailment relation, with higher values indicating higher confidence in entailment. All the relations are ranked based on their score, and Average Precision (AP) is used to evaluate the performance:

$$\text{AP} = \frac{1}{R}\sum_{i=1}^{N}\frac{E(i)\times CorrectUpTo(i)}{i} \tag{5.4}$$

where R is the total number of correct entailment relations, N is the number of possible relations in the test set, $E(i)$ is 1 if the i-th relation is entailment in the gold standard and 0 otherwise, and $CorrectUpTo(i)$ is the number of correctly returned entailment relations up to rank i. Average precision assigns a higher score to systems which rank correct entailment examples higher in the list.

As a secondary measure, we also report the Break-Even Point (BEP), which is defined as precision at the rank where precision is equal to recall. Using the previous annotation, this can also be calculated as precision at rank R:

$$BEP = \frac{CorrectUpTo(R)}{R} \tag{5.5}$$

BEP is a much more strict measure, treating the task as binary classification and ignoring changes to the ranks within the classes.

The test set is balanced, therefore random guessing is expected to achieve an AP and BEP of 50.0% which can be regarded as the simplest (random) baseline. Table 5.1 contains results for two more basic approaches to the task. For the bag-of-words (BOW) system, the score of A entailing B is found as the proportion of lemmatised words in B that are also contained in A:

$$Score_{bow}(A \rightarrow B) = \frac{|\{b|b \in A, B\}|}{|\{b|b \in B\}|} \tag{5.6}$$

We also tested entailment detection when using only the directional distributional similarity between fragments, as is commonly done for calculating similarity between single words. While both of the systems perform better than random, the results are much lower than those for more informed methods. This also indicates that even though there is some lexical overlap between the fragments,

it is not enough to make accurate decisions about the entailment relations.

System type	AP	BEP
Random baseline	50.00	50.0
BOW	65.66	61.0
Distributional similarity (*ClarkeDE*)	64.49	48.0

Table 5.1: Results for the basic approaches on fragment entailment detection.

We were interested in seeing what types of similarity measures perform well for entailment detection between larger fragments, compared to single words. For this experiment, the system was set up to use intrinsic and extrinsic similarity, as described in Section 5.5, but without the additional modifications of paraphrase checking and hedge/negation detection. We utilised the top similarity measures from the hyponym detection task in Chapter 4, and evaluated them as predictors of extrinsic similarity (ExtSim) in the fragment entailment detection task. Different measures were allowed for word-level and fragment-level similarity calculation. Table 5.2 shows a subset of our results, and the full matrix can be found in Appendix C.

The results show that fragment-level entailment detection requires somewhat different measures compared to word-level hyponym detection and generation. Similarity measures such as WeightedCosine and DiceGen2, which were most accurate in previous tasks, do not perform as well here. In contrast, the best results are achieved with ClarkeDE and WeedsPrec, both of which focus even more on the directional nature of the relation, and directly model the subsumption ratio of features. This can be explained by larger text fragments being less ambiguous and occurring in fewer contexts, therefore more closely matching the underlying inclusion hypothesis of these measures. The best performing combination is using ClarkeDE for fragments and LinD for words, and we employ these settings in further experiments.

In order to investigate the contribution of each component to the final system, we conducted an experiment where the components are added incrementally. Table 5.3 shows the results, starting only with the intrinsic similarity between words and fragments.

Using only the intrinsic similarity, the system performs better than any of the

	Cosine	**Lin**	**DiceGen2**	**ClarkeDE**	**LinD**
Cosine	73.21	74.40	73.11	73.31	75.43
Lin	72.72	72.67	70.69	72.84	73.94
JaccardGen	72.64	74.34	74.20	72.35	75.72
DiceGen	73.59	74.74	74.08	73.24	75.89
DiceGen2	71.63	72.92	70.66	71.17	73.98
ClarkeDE	77.45	79.17	79.06	77.28	**80.08**
WeedsPrec	75.82	75.00	72.76	75.97	75.36
WeedsRec	66.52	66.20	62.65	66.54	66.84
WeedsF	72.93	72.62	70.85	72.86	73.94
AP	71.71	72.97	71.42	71.42	73.99
APInc	73.89	75.00	74.90	73.62	76.00
BalAPInc	74.54	75.76	76.24	74.33	76.75
LinD	74.37	73.54	70.92	74.04	73.42
BalPrec	72.84	71.70	71.38	72.87	73.76
WeightedCosine	72.79	73.68	71.58	72.99	74.32

Table 5.2: Average Precision of the entailment detection system using different distributional similarity measures. Rows correspond to the ExtSim measures between fragments, columns represent measures between words.

basic approaches, delivering 70.98% AP. Next, the extrinsic similarity between words is included, raising the AP to 75.43%. When the string-level similarity fails, the added directional distributional similarity helps in mapping semantically equivalent words to each other.

The inclusion of extrinsic similarity between fragments gives a further increase, with AP of 80.08%. The 4.6% increase shows that while fragments are larger and occur less often in a corpus, their distributional similarity can still be used as a valuable component to detect semantic similarity and entailment.

Checking for negation and hedge cues raises the AP to 83.09%. The performance is already high and a 3% improvement shows that hedging and negation affect fragment-level entailment, and other components do not manage to successfully capture this information.

Finally, applying the fragment-level check for paraphrases with a more appropriate distributional similarity measure, as described in Section 5.5.2, returns an AP of 84.14%. The results of this final configuration are significantly different compared to the initial system using only intrinsic similarity, according to the

System type	AP	BEP
Intrinsic similarity only	70.98	68.0
+ Word ExtSim	75.43	71.0
+ Fragment ExtSim	80.08	71.0
+ Negation & hedging	83.09	72.0
+ Paraphrase check	84.14	72.0

Table 5.3: Results for the system described in Section 5.5. Components are added incrementally.

Wilcoxon signed rank test at the level of 0.05.

The formula in Section 5.5 contains parameter α which can be tuned to adjust the balance of intrinsic and extrinsic similarity. This can be done heuristically or through machine learning methods, and different values can be used for fragments and words. In order to investigate the effects of tuning on the system, we tried all possible combinations of α_{word} and $\alpha_{fragment}$ with values between 0.0 and 1.0 at increments of 0.05. Table 5.4 contains results for some of these experiments.[1]

α_{word}	$\alpha_{fragment}$	AP	BEP
0.5	0.5	84.14	72.0
*	0.0	65.61	48.0
0.0	1.0	81.30	72.0
1.0	1.0	76.59	69.0
0.45	0.65	84.75	74.0

Table 5.4: Results of tuning the weights for intrinsic and distributional similarity.

The best results were obtained with $\alpha_{word} = 0.45$ and $\alpha_{fragment} = 0.65$, resulting in 84.75% AP and 74.0% BEP. This shows that parameter tuning can improve the results, but the 0.6% increase is modest, and a completely unsupervised approach can provide competitive results. In addition, the optimal values of α are close to 0.5, indicating that all four components (intrinsic and distributional similarities for words and fragments) are all contributing to the performance of the final system.

[1]The system presented in Table 5.4 uses all the components (checking for negation, hedging and paraphrasing), and therefore only matches Table 5.3 at parameter values 0.5.

5.8 Conclusion

Entailment detection systems are generally developed to work on specific text units – either single words, relations, or full sentences. This reduces the complexity of the problem, but can also lead to important information being disregarded. In this work we proposed a new task – detecting entailment relations between any kind of dependency graph fragments. Dependency graphs provide a natural way of separating complete sentences into smaller units, and our definition of a fragment covers the language structures mentioned above, while also extending to others that have not been fully investigated in the context of entailment recognition (such as multi-word constituents, predicates and adjuncts).

To perform entailment detection between various types of graph fragments, a new system was built that combines the directional intrinsic and extrinsic similarities of two fragments to reach a final score. Fragments which contain hedging or negation are identified and their score calculation is inverted to better model the effect on entailment. The extrinsic similarity is found with two different distributional similarity measures, first checking for symmetric similarity and then for directional containment of distributional features. The system was evaluated on a manually constructed dataset of fragment-level entailment relations from the biomedical domain and each of the added components improved the results.

Traditionally, the method for entailment recognition is chosen based on what appears optimal for the task – either structure matching or distributional similarity. Our experiments showed that the combination of both gives the best performance for all fragment types. It is to be expected that single words will benefit more from distributional measures while full sentences get matched by their components. However, this separation is not strict and evidence from both methods can be used to strengthen the decision.

Our experiments also demonstrate that the distributional hypothesis can be successfully applied to larger graph fragments, but optimal performance is achieved with different measures, compared to working only with single words. As larger fragments are less ambiguous but more sparse, they are more suitable for measures that emphasise the subsumption ratio of context features.

As this is a new task with only a small annotated dataset, we were unable

to run comparative experiments using a supervised learning approach. However, using an unsupervised approach based only on intrinsic and extrinsic similarity, the system was able to achieve 84.14% average precision. The experiments confirmed that entailment between dependency graph fragments of various types can be detected in a completely unsupervised setting, without the need for specific resources or annotated training data. As this method can be applied equally to any domain and requires only a large plain text corpus, we believe it is a promising direction for future research in entailment detection. This can lead to useful solutions in biomedical information extraction where manually annotated datasets are in short supply. The framework for fragment-level entailment detection can be integrated into various applications that require identifying and rewriting semantically equivalent phrases – for example, query expansion in IE and IR, text mining, sentence-level entailment composition, and relation extraction.

Chapter 6

Parser Lexicalisation

6.1 Introduction

In recent years, dependency parsers have become increasingly important tools in the field of natural language processing. They are often used to provide a syntactic and semantic analysis of the input text for tasks such as information extraction, text mining, machine translation, word sense disambiguation, named entity recognition, and coreference resolution. In previous chapters we described some additional innovative ways of utilising automatically generated dependency graphs for language processing. It is necessary to develop robust and accurate parsing methods, as any parser errors are likely to be propagated to other components, impacting the overall performance.

Most parsers make use of machine learning and a syntactically annotated dataset (e.g., treebank), incorporating a wide range of features in the training process to deliver a competitive performance. The use of lexical features, such as lemma or word forms, is very important when choosing the correct derivation in ambiguous contexts. However, this also leads the parser to learn rules that are highly specific to the domain of the training data, and lexical features have been shown to not transfer well between different domains and genres (Sekine, 1997; Gildea, 2001). Furthermore, manual creation of treebanks is an expensive and time-consuming process, which can only be performed by experts with sufficient linguistic and domain knowledge.

In contrast, unlexicalised parsers avoid the use of specific lexical information and aim to provide a syntactic analysis using only more general features, such as POS tags. While they are not expected to achieve the highest accuracy on most well-known treebanked datasets, unlexicalised parsers can be surprisingly competitive with their lexicalised counterparts (Klein & Manning, 2004; Petrov *et al.*, 2006). In this work, instead of trying to adapt a lexicalised parser to other domains, we explore how bilexical features can be integrated with an unlexicalised parser. As our method requires only a large unannotated corpus, lexical features can be easily tuned to a specific domain or genre by selecting a suitable dataset. In addition, our graph expansion process allows the system to capture selected bilexical relations that improve performance but would require sparse higher-order feature types in most dependency parsers. As any bilexical features can still be sparse, we also develop a novel approach to estimating confidence scores for dependency relations using directional distributional similarity measures. The final framework integrates easily with any unlexicalised (and therefore potentially less domain/genre-biased) parser capable of returning ranked dependency analyses.

6.2 Research Goals

Our work on parser lexicalisation has the following research goals:

- Test the hypothesis that lexical features can increase the accuracy of an unlexicalised parser using a self-learning process.
- Investigate methods for automatically extending and modifying dependency graphs, thereby increasing the number of binary relations available for feature extraction.
- Develop novel methods for assigning confidence scores to bilexical relations, using unsupervised corpus-based methods.
- Investigate possible methods for smoothing the relation scores using distributionally similar words.

- Create a working system for improving the performance of an unlexicalised parser by reranking the derivations in a post-processing stage, using only an unannotated text corpus.

6.3 Background

We hypothesise that a large corpus will often contain examples of dependency relations in non-ambiguous contexts, and these will mostly be correctly parsed by an unlexicalised parser. Lexical statistics derived from the corpus can then be used to select the correct parse in a more difficult context. For example, consider the following sentences:

(28) a. Government projects interest researchers.

b. Government raises interest rates.

c. Government projects receive additional funding.

d. Interest rates are increasing.

Noun-verb ambiguities over *projects* and *interest* might erroneously result in the unlexicalised parser returning similar dependency graphs for both *a* and *b*. However, sentences *c* and *d* contain less ambiguous instances of the same phrases and can provide clues to correctly parsing the first two examples. In a large corpus we are likely to find more cases of *researchers* being the object for *interest* and fewer cases where it is the object of *project*. In contrast, *rates* is more likely to have *interest* as a modifier than as a head in an object relation. Exploiting this lexical information, we can assign the correct derivation to each of the more ambiguous sentences.

Similar intuitions have been used to motivate the acquisition of corpus-based bilexical features for improving parser accuracy. However, previous work has only focused on including these statistics as auxiliary features during supervised training of lexicalised parsers. van Noord (2007) incorporates bilexical preferences as features via self-training to improve the Alpino parser for Dutch. The bilexical frequencies are collected from parsed versions of Dutch Wikipedia and

Europarl, mapped to more general feature types using POS tags, converted to pointwise mutual information scores, and then integrated into the parser's maximum entropy parse ranking model. Similarly, Zhou *et al.* (2011) extract n-gram counts from Google queries and a large corpus to improve the MSTParser. They construct a range of features based on lexical co-occurrence and weight them using pointwise mutual information. Plank & van Noord (2008) investigate the application of auxiliary distributions for domain adaptation. They incorporate information from both in-domain and out-of-domain sources into their maximum entropy model and find that the out-of-domain auxiliary distributions do not contribute to parsing accuracy in the target domain.

We formulate our self-learning framework as a reranking process that assigns new scores to top derivations found by the original parser. Parse reranking has been successfully used in previous work as a method of including a wider range of features to rescore a smaller selection of highly-ranked candidate parses. Collins (2000) was one of the first to propose supervised reranking as an additional step to increase parser accuracy and achieved a 1.55% accuracy improvement. Charniak & Johnson (2005) utilise a discriminative reranker and show a 1.3% improvement for the Charniak parser. McClosky *et al.* (2006) extend their work by adding new features and further increase the performance by 0.3%. Ng *et al.* (2010) implemented a discriminative maximum entropy reranker for the C&C parser and showed a 0.23% improvement over the baseline. All of them treat reranking as a supervised task and train a discriminative classifier using parse tree features and annotated in-domain data. In contrast, our reranker uses statistics from an unlabelled source and requires no manual annotation. However, we utilise an unlexicalised parser, therefore, the baseline performance on WSJ text is lower compared to most fully-lexicalised parsers (Briscoe & Carroll, 2006).

Our experiments in Section 6.6 demonstrate that parsing accuracy can be significantly improved by acquiring *all* bilexical features through a self-learning process and utilising them *only* in an unsupervised reranking component. Therefore, the system starts from a simpler unlexicalised parser, less likely to be biased to domains or genres manifested in the training data. The reranker then has the advantage of being able to utilise open-ended amounts of text for lexicalising the parser and tuning it to a required domain and/or genre.

6.4 Reordering dependency graphs

For every sentence s the parser returns a list of dependency graphs G_s, ranked by the log probability of the associated derivation in the structural PGLR model. Our goal is to reorder this list to improve ranking accuracy and, most importantly, to improve the quality of the highest-ranked dependency graph. This is done by assigning a confidence score to every graph $g_{s,r} \in G_s$ where r is the rank of g_s for sentence s. The system treats each sentence independently, therefore we can omit the sentence identifiers and refer to $g_{s,r}$ as g_r.

Our framework first calculates confidence scores for all the individual edges and then combines them into an overall score for the dependency graph. In the following sections we start by describing a series of graph modifications that allows the system to incorporate selected higher-order relations, without introducing unwanted noise or complexity into the reranker. Next, we outline different approaches for calculating and smoothing the confidence score of a bilexical relation. Finally, we describe methods for combining together alternative scores and calculating an overall confidence score for a dependency graph.

6.4.1 Graph modifications

For every dependency graph g_r the system creates a modified representation g'_r which contains a wider range of bilexical relations. The motivation for this graph expansion step is similar to that motivating the move from first-order to higher-order dependency path features (e.g., Carreras, 2007). However, compared to using all second-order paths, these rules are chosen to maximise the utility and minimise the sparsity of the resulting bilexical features. In addition, the cascading nature of the rules means in some cases the expansion can even produce useful 3rd and 4th order dependencies. Similar approaches to graph modifications have been successfully used for several NLP tasks (van Noord, 2007; Arora *et al.*, 2010).

For any edge e we also use notation (rel, w_1, w_2), referring to an edge from w_1 to w_2 with the label rel. Our system performs the following modifications on every dependency graph:

1. **Normalising lemmas.** All lemmas are converted to lowercase. Numerical

lemmas are replaced with more generic tags to reduce sparsity: numbers (e.g., *1*, *104*), lexical representations of numbers (*one*, *eighty*), currency values (*$10,000*, *£20*), enumerative lemmas (*1st*, *2nd*).

2. **Bypassing conjunctions.** For every edge pair (rel_1, w_1, w_2) and (rel_2, w_2, w_3) where w_2 is tagged as a conjunction, we create an additional edge (rel_1, w_1, w_3). This bypasses the conjunction node and creates direct edges between the head and dependents of the conjunctive lemma.

3. **Bypassing prepositions.** For every edge pair (rel_1, w_1, w_2) and (rel_2, w_2, w_3) where w_2 is tagged as a preposition, we create an additional edge (rel_3, w_1, w_3). $rel_3 = rel_1$ + '_prep', where '_prep' is added as a marker to indicate that the relation originally contained a preposition.

4. **Bypassing verbs.** For every edge pair (rel_1, w_1, w_2) and (rel_2, w_1, w_3) where w_1 is tagged as a verb, w_2 and w_3 are both tagged as open-class words, rel_1 starts with a subject relation (*subj*, *ncsubj*, *xsubj*, *csubj*), rel_2 starts with an object relation (*obj*, *dobj*, *iobj*, *obj2*), we create an additional edge (rel_3, w_2, w_3) where $rel_3 = rel_1$ + '-' + rel_2. This creates an additional edge between the subject and the object, with the new edge label containing both of the original labels.

5. **Duplicating nodes.** For every existing node in the graph, containing the lemma and POS for each token (*lemma_pos*), we create a parallel node without the POS information (*lemma*). Next, for each edge *(rel, lemma1_pos1, lemma2_pos2)* we create three additional edges using these new nodes: *(rel, lemma1_pos1, lemma2)*, *(rel, lemma1, lemma2_pos2)*, *(rel, lemma1, lemma2)*. This quadruples the number of edges in each graph and allows the system to use both specific and more generic instantiations of each lemma.

Figure 6.1 illustrates the graph modification process. It is important to note that each of these modifications gets applied in the order that they are described here. For example, when creating edges for bypassing verbs, the new edges for prepositions and conjunctions have already been created and also participate in

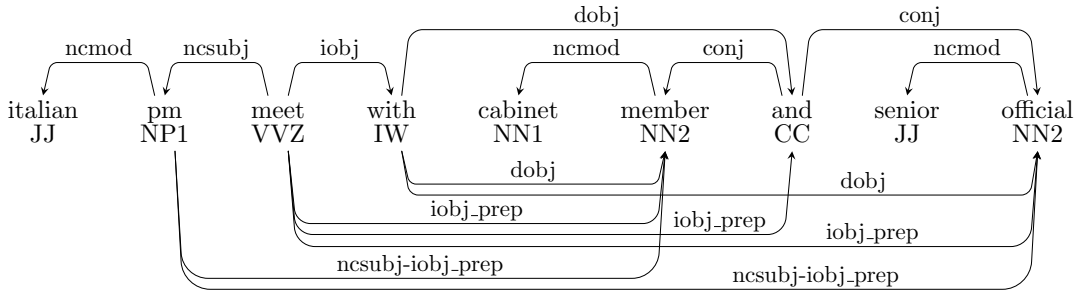

Figure 6.1: Modified graph for the sentence *'Italian PM meets with Cabinet members and senior officials'* after steps 1-4. Edges above the text are created by the parser, edges below the text are automatically created using the operations described in Section 6.4.1. The 5th step will create 9 new nodes and 45 additional edges (not shown).

this operation. We performed ablation tests on the development data and verified that each of these modifications positively contributes to the final performance.

6.4.2 Edge scoring methods

We start the scoring process by assigning individual confidence scores to every bilexical relation in the modified graph. In this section we present some possible strategies for performing this task.

The parser returns a ranked list of graphs and this can be used to derive an edge score without requiring any additional information. We estimate that the likelihood of a parse being the best possible parse for a given sentence is roughly inversely proportional to the rank that it is assigned by the parser. These values can be summed for all graphs that contain a specific edge and normalised to approximate a probability. We then calculate the score for edge e as the Reciprocal Edge Score (RES) – the probability of e belonging to the best possible parse:

$$RES(e) = \frac{\sum_{r=1}^{R}[\frac{1}{r} \times contains(g'_r, e)]}{\sum_{r=1}^{R} \frac{1}{r}} \tag{6.1}$$

where R is the total number of parses for a sentence, and $contains(g'_r, e)$ returns 1 if graph g'_r contains edge e, and 0 otherwise. The value is normalised, so that

an edge which is found in all parses will have a score of 1.0, but occurrences at higher ranks will have a considerably larger contribution.

The score of an edge can also be assigned by estimating the probability of that edge using a parsed reference corpus. van Noord (2007) improved overall parsing performance in a supervised self-training framework using feature weights based on pointwise mutual information:

$$I(e) = log \frac{P(rel, w_1, w_2)}{P(rel, w_1, *) \times P(*, *, w_2)} \tag{6.2}$$

where $P(rel, w_1, w_2)$ is the probability of seeing an edge from w_1 to w_2 with label rel, $P(rel, w_1, *)$ is the probability of seeing an edge from w_1 to any node with label rel, and $P(*, *, w_2)$ is the probability of seeing any type of edge linking to w_2. Plank & van Noord (2008) used the same approach for semi-supervised domain adaptation but were not able to achieve similar performance benefits. In our implementation we omit the logarithm in the equation, as this improves performance and avoids problems with $log(0)$ for unseen edges.

$I(e)$ compares the probability of the complete edge to the probabilities of partially specified edges, but it assumes that w_2 will have an incoming relation, and that w_1 will have an outgoing relation of type rel to some unknown node. These assumptions may or may not be true – given the input sentence, the system has observed w_1 and w_2 but does not know about possible relations they are involved in. Therefore, we create a more general version of this measure that compares the probability of the complete edge to the individual probabilities of the two words – the Conditional Edge Score (CES_1):

$$CES_1(e) = \frac{P(rel, w_1, w_2)}{P(w_1) \times P(w_2)} \tag{6.3}$$

where $P(w_1)$ is the probability of seeing w_1 in text, estimated from a background corpus using maximum likelihood.

Finally, we know that w_1 and w_2 are in a sentence together but cannot assume that there is a dependency relation between them. However, we can choose to think of each sentence as a fully connected graph, with an edge going from every word to every other word in the same sentence. If there exists no genuine relation between the words, the edge is simply considered a *null* edge. We can then find

the conditional probability of the relation type given the two words:

$$CES_2(e) = \frac{P(rel, w_1, w_2)}{P(*, w_1, w_2)} \tag{6.4}$$

where $P(rel, w_1, w_2)$ is the probability of the fully-specified relation, and $P(*, w_1, w_2)$ is the probability of there being an edge of any type from w_1 to w_2, including a *null* edge. Using fully connected graphs, the latter is equivalent to the probability of w_1 and w_2 appearing in a sentence together, which again can be calculated from the background corpus.

6.4.3 Smoothing edge scores

Apart from RES, all the scoring methods from the previous section rely on correctly estimating the probability of the fully-specified edge, $P(rel, w_1, w_2)$. Even in a large background corpus these triples will be very sparse, and it can be useful to find approximate methods for estimating the edge scores.

Based on common smoothing techniques applied to language modelling, the system could back-off to a more general version of the relation. For example, if *(dobj, read, publication)* is not frequent enough, the score could be approximated using the probabilities of *(dobj, read, *)* and *(dobj, *, publication)*. However, this can lead to unexpected results due to compositionality – while *(dobj, read, *)* and *(dobj, *, rugby)* can be fairly common, *(dobj, read, rugby)* is an unlikely relation.

Instead, we can consider looking at other words which are similar to the rare words in the relation. If *(dobj, read, publication)* is infrequent in the data, the system might predict that *book* is a reasonable substitute for *publication* and use *(dobj, read, book)* to estimate the original probability.

Given that the system has a reliable way of finding likely substitutes for a given word, we can create expanded versions of CES_1 and CES_2, as shown in Equations 6.5 and 6.6. C_1 is the list of substitute words for w_1, and $sim(c_1, w_1)$ is a measure showing how similar c_1 is to w_1. The methods iterate over the list of substitutes and calculate the CES score for each of the modified relations. The values are then combined by using the similarity score as a weight – more similar words will have a higher contribution to the final result. This is done for both the head and the dependent in the original relation, and the scores are then

normalised and averaged.

$$ECES_1(rel, w_1, w_2) = \frac{1}{2} \times \left(\frac{\sum_{c_1 \in C_1} sim(c_1, w_1) \times \frac{P(rel, c_1, w_2)}{P(c_1) \times P(w_2)}}{\sum_{c_1 \in C_1} sim(c_1, w_1)} + \frac{\sum_{c_2 \in C_2} sim(c_2, w_2) \times \frac{P(rel, w_1, c_2)}{P(w_1) \times P(c_2)}}{\sum_{c_2 \in C_2} sim(c_2, w_2)}\right) \tag{6.5}$$

$$ECES_2(rel, w_1, w_2) = \frac{1}{2} \times \left(\frac{\sum_{c_1 \in C_1} sim(c_1, w_1) \times \frac{P(rel, c_1, w_2)}{P(*, c_1, w_2)}}{\sum_{c_1 \in C_1} sim(c_1, w_1)} + \frac{\sum_{c_2 \in C_2} sim(c_2, w_2) \times \frac{P(rel, w_1, c_2)}{P(*, w_1, c_2)}}{\sum_{c_2 \in C_2} sim(c_2, w_2)}\right) \tag{6.6}$$

We use *WeightedCosine*, the best-performing directional distributional similarity measure from Chapter 4, to generate a list of hyponyms for substituting each word in the relation. Hyponyms are well-suited for lexical expansion, as the properties of a hypernym (e.g., *publication*) are also expected to apply to its hyponyms (*book*, *magazine*, *catalogue*). In addition, by utilising a distributional similarity measure we are able to directly use the real-valued similarity score in the equations and avoid the use of any domain-specific or manually annotated resources.

In order to calculate word similarity scores, the feature vectors for each word are built from the same vector space model that is used for calculating edge probabilities, which includes all the graph modifications described in Section 6.4.1. Features are either *(rel, head)* or *(rel, dependent)* tuples, corresponding to the directed relations that the word is involved in, weighted by mutual information. The system calculates the similarity between a word and all other words in the corpus, retaining the top 10 highest ranking substitutes. We also include the original word in this list with similarity 1.0, thereby assigning the highest weight to the original relation.

6.4.4 Combining edge scores

While the CES and $ECES$ measures calculate confidence scores for bilexical relations using statistics from a large background corpus, they do not include any knowledge about grammar, syntax, or the context in a specific sentence. In contrast, the RES score implicitly includes some of this information, as it is calculated based on the original parser ranking. We wish to combine these scores, in order to take advantage of both information sources. Since the means and variances of alternative scoring methods can be rather different, we found that taking the geometric mean gives the best performance. In Section 6.6 we evaluate the following two combination scores:

$$CMB_1(e) = \sqrt[3]{RES(e) * CES_1(e) * CES_2(e)} \tag{6.7}$$

$$CMB_2(e) = \sqrt[3]{RES(e) * ECES_1(e) * ECES_2(e)} \tag{6.8}$$

6.4.5 Graph scoring

Every edge in graph g'_r is assigned a score indicating the reranker's confidence in that edge belonging to the best parse. We investigated different strategies for combining these values together into a confidence score for the whole graph. The simplest solution is to sum together individual edge scores, but this would lead the system to always prefer graphs that have a larger number of edges. Interestingly, averaging the edge scores does not produce good results either – the system will be biased towards smaller graph fragments containing only highly-confident edges.

We created a new scoring method which prefers graphs that connect all the nodes, yet does not explicitly bias towards a higher number of edges. For every node in the graph, it finds the average score of all edges which have that node as a dependent. These scores are then averaged again over all nodes:

$$NodeScore(n) = \frac{\sum_{e \in E_g} EdgeScore(e) \times isDep(e, n)}{\sum_{e \in E_g} isDep(e, n)} \tag{6.9}$$

$$GraphScore(g) = \frac{\sum_{n \in N_g} NodeScore(n)}{|N_g|} \tag{6.10}$$

where g is the graph being scored, N_g is the set of nodes in g, E_g is the set of edges in g, $n \in N_g$ is a node in graph g, $e \in E_g$ is an edge in graph g, $isDep(e, n)$ is a function returning 1.0 if n is the dependent in edge e, and 0.0 otherwise. $NodeScore(n)$ is set to 0 if the node does not appear as a dependent in any edges. We found this method to perform well, as it prefers graphs that connect together many nodes without simply rewarding a larger number of edges. However, future research could be directed towards integrating the edge scores more closely with the underlying parsing algorithm, potentially leading to improved accuracy.

It is important to note that while the score calculation is done using the modified graph, the resulting score is directly assigned to the corresponding original graph, and the reordering of the original dependency graphs is used for evaluation.

6.5 Evaluation methods

In order to evaluate how much the reranker improves the highest-ranked dependency graph, we calculate the microaveraged precision, recall and F-score over all dependencies from the test and gold standard parses. Following the official RASP evaluation (Briscoe *et al.*, 2006) we employ the hierarchical edge matching scheme which aggregates counts up the GR subsumption hierarchy and thus rewards the parser for making more fine-grained distinctions.[1]

Statistical significance of the change in F-score is calculated using the Approximate Randomisation Test (Noreen, 1989; Cohen, 1995) with 10^6 iterations. This method is a computer-intensive statistical hypothesis test which can be used for many NLP tasks (Yeh, 2000), including parsing (Cahill *et al.*, 2008). It is designed to assess result differences with respect to a test statistic in cases where the sampling distribution of the test statistic is unknown.

Under the null hypothesis, the new system is not different from the baseline,

[1] Slight changes in the performance of the baseline parser compared to previous publications are due to using a more recent version of the parser and minor corrections to the gold standard annotation.

therefore any analysis produced by one of these systems is just as likely to have been produced by the other. This is tested by switching dependency graphs between the two systems and recomputing the difference between the evaluation metrics. For n sentences there are 2^n different ways of shuffling the graphs, which can get computationally intractable for even a relatively small n. Instead of generating every possible combination, the test can be performed using an approximate randomisation over many iterations, where each shuffle is performed with random assignments – for every sentence, there is a 50% probability of switching the corresponding dependency graphs from both systems. We count the number of times c when the difference between the F-scores of two randomly shuffled outputs is greater or equal than the difference between our system and the baseline. The probability of obtaining a difference in the evaluation metric at least as extreme as the one that was actually observed, assuming that the null hypothesis is true, is then calculated as:

$$p = \frac{c+1}{R+1} \tag{6.11}$$

where R is the number of trials performed (Noreen, 1989). If the value is smaller than 0.05, the null hypothesis is rejected and the systems are considered significantly different. For experiments in Section 6.6 we use $R = 10^6$.

We also wish to measure how well the reranker does at the overall task of ordering dependency graphs. For this we make use of an oracle that creates the perfect ranking for a set of graphs by calculating their individual F-scores; this ideal ranking is then compared to the output of our system. Spearman's rank correlation coefficient between the two rankings is calculated for each sentence and then averaged over all sentences. If the scores for all of the returned analyses are equal, this coefficient cannot be calculated and is set to 0.

Finally, we also report the number of times the system assigns the highest score to the best available dependency graph.

6.6 Experiments

6.6.1 DepBank

The system was evaluated using the DepBank/GR reannotation (Briscoe & Carroll, 2006) of the PARC 700 Dependency Bank (King *et al.*, 2003), described in Section 2.3.7. The dataset is provided with the open-source RASP distribution and has been used for evaluating different parsers, including RASP (Briscoe & Carroll, 2006; Watson *et al.*, 2007) and C&C (Clark & Curran, 2007). It contains 700 sentences, randomly chosen from section 23 of the WSJ Penn Treebank (Marcus *et al.*, 1993), divided into development (140 sentences) and test data (560 sentences).

We made use of the development data to experiment with a wider selection of edge and graph scoring methods. However, we found that the results between different configurations became more stable after removing the 10 longest sentences in the development set. Long and convoluted sentences are most difficult for the parser and reranker to handle correctly. At the same time, they contribute a large proportion of the final results, as the dataset is very small and the F-score is averaged over individual dependencies, not sentences. Therefore, we used 130 sentences to perform experiments during the development process.

For reranking we collect up to 1,000 top-ranked parses for each sentence. The actual number of parses that RASP outputs depends on the sentence and can be smaller. As RASP first constructs parse trees and converts them to dependency graphs, different parses of the same sentence may result in identical graphs; we remove any duplicates to obtain a ranking of unique dependency graphs.

The reranking framework relies on a large unannotated corpus of in-domain text, and for this we used the BLLIP corpus containing 50M words of in-domain WSJ articles. Our version of the corpus excludes texts that are found in the Penn Treebank, thereby also excluding the section that we use for evaluation. All the data was reparsed with RASP, the resulting highest-ranked dependency graphs were modified using the process described in Section 6.4.1, and the required occurrence counts for words and relations were automatically collected.

The baseline system is the unlexicalised RASP parser with default settings. In

DepBank/GR Development Set (130)

	Prec	**Rec**	**F**	ρ	**Best**
Baseline	77.90	74.43	76.13	33.81	49
Upper Bound	87.29	83.09	85.14	76.15	130
I	76.93	73.80	75.33	31.25	54
RES	77.23	73.77	75.46	39.26	47
CES_1	78.34	74.90	76.58	35.48	56
CES_2	79.61	76.06	77.79	43.72	54
$ECES_1$	78.50	75.10	76.77	38.23	55
$ECES_2$	80.06	76.73	78.36	44.49	**57**
CMB_1	79.67	76.04	77.81	43.95	**57**
CMB_2	**80.24**	**76.66**	**78.41**	**45.95**	55

Table 6.1: Performance of different edge scoring methods on the development data. For each measure we report precision, recall, F-score, average Spearman's correlation (ρ), and the number of times the system assigns the highest rank to the best available dependency graph (Best). The highest results for each measure are marked in bold. The underlined F-scores are significantly better compared to the baseline.

order to identify an upper bound, we use an oracle to calculate the F-score for each dependency graph individually, and then create the best possible ranking using these scores. The upper bound for precision, recall and F-measure is less than 100%, as the parser does not always return a perfect derivation as one of the top 1000 candidates. Average Spearman's correlation for the upper bound is expected to be 1.0, as it is compared to itself. However, when all the dependency graphs have the same score according to the gold standard (including when the parser returns only 1 analysis), then the correlation coefficient cannot be calculated and is, therefore, set to 0.

Tables 6.1 and 6.2 contain evaluation results on the development and test sets. The baseline system achieves 76.41% F-score on the test data, with 32.70% average correlation, and returns the best possible parse as the top parse 231 times out of possible 560. I and RES scoring methods give results comparable to the baseline, with RES improving correlation by 9.56%. The CES scores make use of corpus-based statistics and significantly improve over the baseline system. The F-score is increased to 78.85% and additional 39 sentences are correctly ranked as the top parses. The $ECES$ scores utilise similarity-based smoothing and improve

DepBank/GR Test Set (560)

	Prec	Rec	F	ρ	Best
Baseline	77.91	74.97	76.41	32.70	231
Upper Bound	86.74	82.82	84.73	75.36	560
I	77.77	75.00	76.36	33.32	238
RES	78.13	74.94	76.50	42.26	228
CES_1	79.68	76.40	78.01	41.95	262
CES_2	80.48	77.28	78.85	48.43	270
$ECES_1$	79.96	76.68	78.29	42.41	266
$ECES_2$	80.71	77.52	79.08	**49.05**	272
CMB_1	80.64	77.31	78.94	48.25	263
CMB_2	**80.88**	**77.60**	**79.21**	49.02	**273**

Table 6.2: Performance of different edge scoring methods on the test data.

the results even further, giving the F-score of 79.08%.

Finally, we combine the *RES* score with the corpus-based methods by finding their geometric mean, and the CMB_2 measure delivers the best overall results. The final F-score is 79.21%, an improvement of 2.8%, corresponding to 33.65% relative error reduction with respect to the upper bound. Correlation is increased by 16.32% and 273 sentences get assigned the correct top analysis. This means the methods not only manage to create a better overall ranking, but also improve the chances of differentiating between the best dependency graph and all the others. The F-scores for all the corpus-based scoring methods are statistically significant compared to the baseline ($p < 0.05$).

By using only a plain-text corpus and a self-learning framework, the system was able to significantly improve the original unlexicalised parser. To put the overall result in wider perspective, Clark & Curran (2007) achieve an F-score of 81.86% on the DepBank/GR test sentences using the C&C lexicalised parser, trained on 40,000 manually-treebanked sentences from the WSJ. The RASP unlexicalised parser, using a manually-developed grammar and a parse ranking component trained on 4,000 partially-bracketed unlabelled sentences from a domain/genre balanced subset of Brown (Watson *et al.*, 2007), achieves an F-score of 76.41% on the same test set. The method introduced here improves this to 79.21% F-score without using any further manually-annotated data, closing more than half of the gap between the performance of a fully-supervised in-domain

Genia-GR Dataset (492)

	Prec	Rec	F	ρ	Best
Baseline	79.91	78.86	79.38	36.54	**229**
Upper Bound	86.33	84.71	85.51	78.66	492
I	77.18	76.21	76.69	30.23	175
RES	80.06	78.89	79.47	**47.52**	228
CES_1	78.64	77.50	78.07	36.06	192
CES_2	79.92	78.92	79.42	43.09	194
$ECES_1$	79.09	78.11	78.60	38.02	194
$ECES_2$	79.84	78.95	79.39	43.64	195
CMB_1	80.60	79.51	80.05	44.96	204
CMB_2	**80.69**	**79.64**	**80.16**	46.24	206

Table 6.3: Performance of different edge scoring methods on the GENIA-GR dataset.

parser and a semi-supervised more domain-neutral one.

6.6.2 Genia

The strength of our reranking framework is that it does not require any domain-dependent resources or manually annotated training data. Therefore, we are interested in seeing how it performs on a dataset from a completely different domain.

We evaluate this system on the GENIA-GR dataset (Tateisi *et al.*, 2008), a collection of 492 sentences from biomedical research papers in GENIA (Kim *et al.*, 2003). The sentences have been manually annotated with dependency-based grammatical relations that correspond to the same scheme used by RASP. However, it does not contain dependencies for all tokens and many multi-word phrases are treated as single units. For example, the tokens *'intracellular redox status'* are annotated as one node with label *intracellular_redox_status*. When parsing, we retain this annotation and allow RASP to treat these nodes as unseen words; however, we use the last lemma in each multi-word phrase to calculate the edge scoring statistics. The dataset is described in more detail in Section 2.3.14.

In order to initialise our parse reranking framework, we also need a background corpus that closely matches the evaluation domain. The annotated sentences in

GENIA-GR were chosen from abstracts that are labelled with the MeSH term *'NF-kappa B'*.[1] Following this method, we created our background corpus by extracting 7,100 full-text articles (1.6M sentences) from PubMed Central Open Access collection, containing any of the following terms with any capitalisation: *'nf-kappa b'*, *'nf-kappab'*, *'nf kappa b'*, *'nf-kappa_b'*, *'nf-kb'*, *'nf-κb'*. Since we retain all texts from matching documents, this keyword search acts as a broad indicator that the sentences contain topics which correspond to the evaluation dataset. This focussed corpus was then parsed with RASP and used to create a statistical model for the reranking system, following the same methodologies as described in Sections 6.4 and 6.6.1.

Table 6.3 contains the results for experiments in the biomedical domain. The first thing to notice is that while the upper bound for RASP is similar to the DepBank experiments in Section 6.6.1, the baseline results are considerably higher. This is largely due to the nature of the dataset – since many complicated multi-word phrases are treated as single nodes, the parser is not evaluated on edges within these nodes. In addition, treating these nodes as unseen words eliminates many incorrect derivations that would otherwise split the phrases. This results in a naturally higher baseline of 79.38%, and also makes it more difficult to further improve the performance.

The edge scoring methods I, CES_1 and $ECES_1$ deliver F-scores lower than the baseline in this experiment. RES, CES_2 and $ECES_2$ manage to give a modest improvement in both F-score and Spearman's correlation. Finally, the combination methods again give the best performance, with CMB_2 delivering an F-score of 80.16%. This is an absolute increase of 0.78% on an already high baseline, and the result is also statistically significant ($p < 0.05$). The experiments show that our self-learning framework works on very different domains, and it can be used to significantly increase the accuracy of an unlexicalised parser without requiring any annotated data.

		Baseline			CMB_2			
Label	**#GRs**	**Prec**	**Rec**	**F**	**Prec**	**Rec**	**F**	**ΔF**
dependent	10682	80.01	77.29	78.62	82.48	79.70	81.07	2.45
aux	400	93.40	92.00	92.70	91.21	90.75	90.98	-1.72
conj	595	74.74	71.60	73.13	76.47	74.29	75.36	2.23
ta	291	41.25	47.77	44.27	42.49	50.52	46.15	1.89
det	1114	87.95	90.39	89.15	89.26	91.74	90.48	1.33
arg_mod	8282	79.22	75.31	77.21	82.36	78.11	80.18	2.96
mod	3906	74.50	67.54	70.85	77.35	70.92	73.99	3.15
ncmod	3548	75.84	69.73	72.66	78.57	73.25	75.82	3.16
xmod	178	50.89	48.31	49.57	56.80	53.93	55.33	5.76
cmod	168	52.58	30.36	38.49	56.52	30.95	40.00	1.51
pmod	12	30.77	33.33	32.00	33.33	33.33	33.33	1.33
arg	4376	77.94	76.76	77.34	81.42	79.09	80.24	2.89
subj_dobj	3120	82.73	75.22	78.80	85.30	77.53	81.23	2.43
subj	1359	81.36	67.77	73.95	84.44	69.46	76.22	2.28
ncsubj	1350	81.95	67.93	74.28	85.05	69.56	76.53	2.25
xsubj	7	33.33	28.57	30.77	33.33	28.57	30.77	0.00
csubj	2	14.29	50.00	22.22	25.00	100.00	40.00	17.78
comp	3017	75.47	79.45	77.41	79.30	82.27	80.75	3.34
obj	2325	78.27	79.96	79.11	80.47	83.14	81.79	2.68
dobj	1761	81.99	79.39	80.67	84.52	82.45	83.47	2.80
obj2	20	20.69	30.00	24.49	27.78	25.00	26.32	1.83
iobj	544	71.09	77.76	74.28	71.91	82.35	76.78	2.50
clausal	668	62.37	72.46	67.04	70.25	74.25	72.20	5.16
xcomp	380	77.08	77.89	77.49	80.16	78.68	79.42	1.93
ccomp	288	47.45	64.58	54.71	58.56	67.71	62.80	8.10
pcomp	24	66.67	66.67	66.67	77.27	70.83	73.91	7.25

Table 6.4: System results on the test data, separated for each label in the GR type hierarchy. The table shows the number of times the label occurs in the gold standard, Precision/Recall/F-score of both the baseline and the best reranking method, and the difference in F-scores between the two systems.

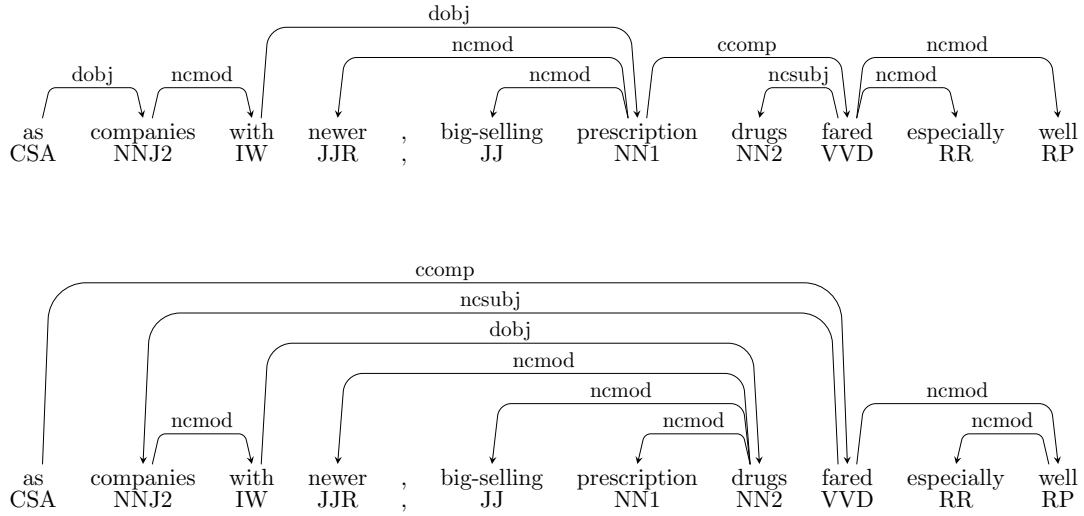

Figure 6.2: Subset of dependency relations for the sentence *'Earnings for most of the nation's major pharmaceutical makers are believed to have moved ahead briskly in the third quarter, as companies with newer, big-selling prescription drugs fared especially well'*. The top graph is ranked highest by the baseline parser; the bottom graph is chosen by the reranker.

6.6.3 Error analysis

Following Briscoe *et al.* (2006) and Clark & Curran (2007), we also present a more fine-grained analysis of the system performance on the DepBank/GR test dataset. Table 6.4 contains the results separated for each label in the GR type hierarchy. The auxiliary dependency is the only relation for which the reranking process reduces both precision and recall in this dataset; all other dependency types show improved results. Complements and modifiers are attached with much higher accuracy, resulting in 3.34% and 3.15% increase in corresponding F-scores. The system has also given an increase of 3.16% for the non-clausal modifier relation (*ncmod*), which is the most frequent label in the dataset. *subj* and *obj* relations show a slightly lower increase, mostly because the baseline system already attaches them with higher accuracy. In contrast, the *obj2* relation retains a rather low accuracy, and this is due to errors in the top 1,000 derivations – the upper bound for the F-score of *obj2* was found to be only 36.84%.

[1]NF-κB is a protein complex found in many mammals, including humans.

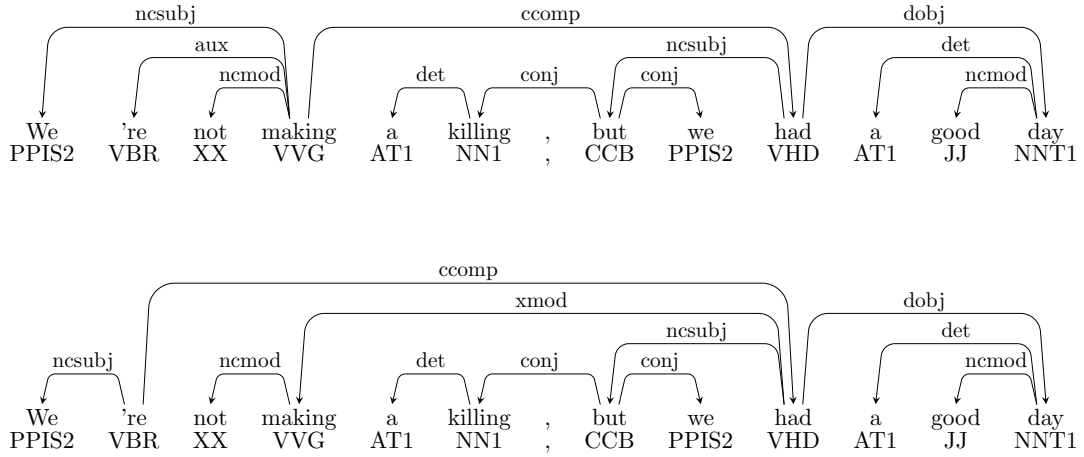

Figure 6.3: Subset of dependency relations for the sentence *'We're not making a killing, but we had a good day.'*. The top graph is ranked highest by the baseline parser; the bottom graph is chosen by the reranker.

We also performed a manual analysis of how the introduction of lexical features has changed the results of the unlexicalised parser. Sentences from the development data were ranked by the unlexicalised parser and by the best reranker configuration, and the output was compared in order to better understand the system differences. Lexicalisation was found to have the most noticeable effect on correctly attaching modifiers, especially in ambiguous contexts. Figure 6.2 contains example graphs from the two systems, demonstrating the improved accuracy of the reranker. Among other things, the system has successfully learned that *companies* are more likely to be the subject of *fare*, and that *especially* is more likely to modify *well*.

In contrast, Figure 6.3 shows an example where the reranking process decreased the accuracy of the top parse. While RASP had correctly attached *are* as an auxiliary verb to *making*, the reranker incorrectly preferred *we* as the subject for *are*. This also illustrates the overall decrease in accuracy for the auxiliary dependency relations. Neither system correctly captured the phrase *making a killing*.

6.7 Conclusion

The use of lexical features, such as lemmas or word forms, is very important for accuracte parsing and can help the system choose the correct derivation in ambiguous contexts. However, it also leads parsers to learn rules that are highly specific to the domain of the training data, and lexical features have been shown to not transfer well between different domains and genres. Furthermore, manual creation of annotated treebanks is an expensive and time-consuming process, which can only be performed by experts with sufficient linguistic and domain knowledge.

Instead of trying to adapt a lexicalised parser to other domains, we explored how bilexical features can be integrated with an unlexicalised parser, without requiring training data. We hypothesised that a large corpus will often contain examples of dependency relations in non-ambiguous contexts, and these will mostly be correctly parsed by an unlexicalised parser. Lexical statistics derived from the corpus can then be used to select the correct parse in a more difficult context.

In this chapter we developed a self-learning framework for dependency graph reranking that requires only a plain-text corpus of in-domain text. We automatically parse this corpus with RASP and use it to estimate maximum likelihood probabilities for bilexical relations. Every dependency graph is first modified to incorporate additional edges that model selected higher-order relationships. Each edge in the graph is then assigned a confidence score based on statistics from the background corpus and preferences from the parser. We also described a novel method for smoothing these scores using directional distributional similarity measures. Finally, the edge scores were combined into an overall graph score by first averaging them over individual nodes.

We performed evaluation on data from two different domains: the DepBank/GR dataset of WSJ text, and the GENIA-GR dataset of biomedical abstracts. On DepBank/GR the reranking process was able to increase the F-score from 76.41% to 79.21%. The improvement was consistent on nearly all individual dependency types, with important benefits for correctly attaching complements and modifiers. On GENIA-GR the corresponding F-score was increased from

79.38% to 80.16%. The lower relative improvement can be partly attributed to an overall higher baseline performance, due to the nature of the dataset. In both experiments the results were found to be statistically significant.

While fully-supervised parsers, such as the C&C parser, still return higher performance on WSJ text, our self-learning framework successfully improved the performance of the unlexicalised RASP parser and reduced the relative difference by 51.4%. As it requires no annotated data, it can be easily adapted to different domains and genres, and this was demonstrated by experiments using biomedical texts. It remains as part of future work to evaluate the performance of WSJ-specific supervised parsers on the GENIA-GR dataset.

Chapter 7

Conclusion

The amount of available textual resources is constantly growing with a remarkable speed, and the field of Natural Language Processing (NLP) offers methods for efficiently organising, analysing and summarising large amounts of information. Many systems with state-of-the-art performance employ supervised machine learning techniques, using a manually labelled dataset to optimise model parameters. This allows them to learn complex patterns from the data, but also makes them highly reliant on manually annotated corpora, which are time-consuming and costly to create. Furthermore, systems trained on domain-specific datasets often do not generalise well to other genres and domains.

In this work we investigated minimally-supervised methods for solving NLP tasks, and we defined them as requiring no explicit training data. However, since completely unsupervised systems are usually only applicable to tasks that can be formulated as clustering problems, we expand this definition by allowing minimal external information to be incorporated. For example, this can be in the form of selected heuristic rules, linguistic knowledge encoded into the system, or by utilising some task-independent tools (e.g., an existing dependency parser). The motivation for our work is to create systems that require substantially reduced effort from domain and/or NLP experts, compared to annotating a corresponding dataset, and also offer easier domain adaptation and better generalisation properties.

We applied the principles of minimal supervision to four NLP tasks and compared their performance to alternative methods:

In Chapter 3 we described a system for detecting the scope of speculation in sentences, by using only 8 manually-defined rules over dependency graphs. We found that the small number of manual rules were able to offer competitive results without requiring any annotation. In addition, the best performance was achieved when the rule-based output was included as an extra feature in a supervised classifier.

Chapter 4 presented experiments for detecting and generating hyponyms using a wide range of distributional similarity measures. We also proposed a new similarity measure which delivered the best performance for hyponym generation, even outperforming a supervised alternative. In contrast, the best results for the hyponym detection task were achieved by including all the similarity measures as features in a classifier.

In Chapter 5 we proposed a new task of detecting entailment relations between dependency graph fragments of various types and sizes. Our system made use of the lexical and distributional similarity of graph fragments and was able to perform entailment detection in a completely unsupervised setting, without the need for specific resources or annotated training data. We were also able to demonstrate that the distributional hypothesis can be applied to larger graph fragments for unsupervised similarity detection.

Finally, in Chapter 6 we described a self-learning framework for improving the accuracy of an unlexicalised parser, by calculating relation probabilities from its own dependency output. The method requires only a large in-domain text corpus and can therefore be easily applied to different domains and genres. Our experiments showed significant improvements on both WSJ and biomedical text, and the system greatly decreased the performance gap between the unlexicalised RASP parser and the lexicalised C&C parser.

While fully supervised approaches generally achieve the highest results, our experiments found minimally supervised methods to be remarkably competitive. For example, by utilising a minimally-supervised system for detecting speculation cues and scopes, the relative F-score was only 3.32% lower, compared to using fully supervised classifiers. At the task of parse reranking, the relative F-score for our self-learning framework was lower by only 3.24%, when compared to the supervised C&C parser. Furthermore, the best distributional similarity measures

were able to outperform a supervised classifier on the task of hyponym generation.

In order to further improve the performance of supervised systems, a common strategy is to provide more data. This also applies for minimally-supervised methods that rely on similarity scores or corpus statistics, and obtaining plain text is considerably easier compared to annotating new examples. In both cases, however, the improvements are expected to reach a plateau as even more data is added. In contrast, rule-based systems can be further improved by examining samples from the system output and iteratively adjusting the rules.

The results presented here are the outcome of four case studies and are unlikely to generalise to every NLP problem. In most tasks, supervised systems consistently deliver the highest performance, and it remains to be seen whether minimally-supervised methods can achieve a similar level. However, by developing unsupervised and minimally-supervised systems, we aim to create solutions that are more likely to generalise across domains and genres. As part of future work, we would encourage development of systems that are capable of taking full advantage of the large amounts of unannotated text available from various sources on the Web. In addition, as dependency graphs can offer an automated method for resolving the structure and semantics in unlabelled text, it is important to further improve the accuracy and versatility of existing parsing methods.

Supervised classifiers can offer out-of-the-box approaches for solving many NLP challenges with relatively high accuracy, but the cost of manually labelling a separate dataset for each different task and domain is likely to be high. By moving away from explicit supervision, we aim to better understand the underlying patterns in the data, and to create systems that are not tied to any specific domains. We described different methods for applying minimal supervision, and showed that these systems are able to deliver competitive performance without any annotation costs. In addition, when manually labelled data is available, these techniques can be used as discriminative features in a combined system to further improve the accuracy. We conclude that minimally supervised methods are viable solutions to many different natural language processing tasks, offering low costs, fast performance, and competitive results.

Appendix A: Hedge cues

Lemma	Cue	Total	Lemma	Cue	Total
suggest	683	685	possibility	15	40
may	583	587	assume	14	25
indicate that	316	319	not know	13	18
appear	182	217	perhaps	13	15
or	146	1215	presumably	12	12
whether	122	129	expect	10	34
might	112	112	predict	10	137
likely	96	101	apparently	9	22
could	95	196	speculate	9	10
possible	79	97	and/or	8	60
can	64	506	estimate	8	58
potential	51	136	not clear	8	8
think	49	52	raise the possibility	8	8
putative	47	69	apparent	7	33
seem	45	45	assumption	7	14
propose	44	64	consider	7	93
either	36	206	not	6	1409
would	31	37	suspect	6	7
possibly	30	30	whether or not	6	6
unknown	29	74	suggestion	5	5
probably	28	29	support	5	90
should	27	37	unlikely	5	5
imply	26	27	idea	4	11
indicate	25	421	notion	4	9
potentially	25	28	cannot be exclude	3	3
hypothesis	22	50	clear	3	39
unclear	20	23	indication	3	5
hypothesize	18	20	infer	3	8
believe	16	18	must	3	25
if	16	114	consistent with	2	76

Table 1: Part 1/2 of all lemmatised speculation cues in the CoNLL-10 training data, including counts of how many times they occurred as cues and in total.

Lemma	Cue	Total	Lemma	Cue	Total
elucidate	2	24	likelihood	1	105
favor	2	8	look as	1	1
hope	2	2	look like	1	1
hypothesise	2	2	may , or may not	1	1
hypothetically	2	2	must be leave open	1	1
if not	2	2	no clear evidence	1	1
implicate	2	73	no evidence of	1	2
no evidence	2	7	no guarantee	1	1
postulate	2	11	no proof	1	2
prediction	2	101	not be clearly elucidate	1	1
probable	2	3	not clearly delineate	1	1
remain to be elucidate	2	3	not evident	1	3
suggestive	2	2	not exclude	1	2
suppose	2	3	not fully understand	1	4
tend	2	8	not possible to ascertain	1	1
address a number of question	1	1	obscure	1	5
address the question of	1	1	plausible	1	1
advance the hypothesis	1	1	potent	1	96
argue	1	5	preferentially	1	36
can be deduce	1	1	presume	1	3
cannot	1	47	prone to	1	2
cannot assign	1	1	proposal	1	1
cannot be	1	25	putatively	1	1
cannot exclude	1	2	question	1	17
cannot exclude the possibility	1	1	raise an interest question	1	1
cannot rule out the possibility	1	1	raise question	1	1
compel evidence	1	2	raise the hypothesis	1	1
conceivable	1	1	raise the intrigue possibility	1	1
conclusion	1	60	raise the question	1	1
elusive	1	3	reason	1	5
evaluate for	1	2	refer to	1	39
examine for	1	10	remain to be determine	1	1
feasible	1	4	remain to be investigate	1	1
feel	1	1	speculation	1	1
happen	1	1	test	1	154
hypothetical	1	8	to confirm	1	7
increase evidence	1	5	uncertain	1	3
investigate	1	237	unproven	1	1
issue be raise	1	1	viable	1	9
know	1	252	with certainty	1	1
lend strong support	1	1	yet to be understand	1	1

Table 2: Part 2/2 of all lemmatised speculation cues in the CoNLL-10 training data, including counts of how many times they occurred as cues and in total.

Appendix B: Hyponym generation examples

We present here some examples of hyponym generation, using a model trained on BNC and the best-performing distributional similarity measures from Chapter 4. Correct hyponyms, according to WordNet, are marked in bold.

scientist (noun)

Cosine	**researcher**, **psychologist**, expert, writer, **chemist**, **biologist**, official, teacher, professional, **sociologist**, practitioner, student, **economist**, journalist, politician
DiceGen2	**researcher**, expert, **psychologist**, writer, official, **economist**, journalist, historian, **chemist**, scholar, analyst, politician, professional, practitioner, engineer
BalAPInc	**biologist**, **sociologist**, **researcher**, **geologist**, **astronomer**, **psychologist**, clinician, **linguist**, **archaeologist**, **chemist**, hydrogeologist, academic, **geneticist**, **investigator**, **anthropologist**
ClarkeDE	medicinal, g.d, ying, endocytosis, battison, chouilly, oakbrook, wissenland, assyria, saurischian, nation-building, thuringia, tidesman, mordor, lawren
BalPrec	**researcher**, **sociologist**, **biologist**, **psychologist**, **investigator**, practitioner, **astronomer**, **chemist**, professional, academic, **archaeologist**, **geologist**, counsellor, commentator, planner
WeightedCosine	**researcher**, expert, **chemist**, **psychologist**, writer, **biologist**, official, **economist**, science, teacher, student, journalist, professional, scholar, historian

politician (noun)	
Cosine	government, official, leader, people, party, scientist, journalist, critic, member, **candidate**, writer, voter, teacher, unionist, businessman
DiceGen2	official, journalist, critic, leader, businessman, scientist, **candidate**, voter, government, activist, writer, lawyer, politics, minister, **democrat**
BalAPInc	unionist, educationalist, proponent, sociologist, protestant, clinician, republicanism, nationalist, diplomat, yorkshiremen, environmentalist, assemblyman, intellectual, **legislator**, **backbencher**
ClarkeDE	medicinal, ying, chouilly, g.d, oakbrook, sidmouth, wissenland, zelan, macbryde, blacksell, ibaez, to-ing, endocytosis, phinehas, evangelicals
BalPrec	unionist, sociologist, participant, delegate, voter, protestant, professional, academic, reformer, commentator, catholic, nationalist, diplomat, activist, researcher
WeightedCosine	government, official, leader, party, critic, people, journalist, member, politics, **candidate**, scientist, businessman, writer, minister, voter

sport (noun)	
Cosine	game, **football**, club, **tennis**, news, **rugby**, **cricket**, race, **racing**, **athletics**, **baseball**, event, television, **golf**, newspaper
DiceGen2	game, **football**, **tennis**, news, race, **racing**, **cricket**, **rugby**, tournament, club, **golf**, championship, newspaper, television, competition
BalAPInc	**baseball**, **archery**, **athletics**, canoeing, curtain-raiser, specialty, **boxing**, **tennis**, troon, all-rounders, soldiering, dramatics, story-telling, adida, g-mex
ClarkeDE	medicinal, oakbrook, thuringia, mordor, orkneys, chouilly, zenda, window-seat, cygnus, sidmouth, richmondshire, kesteven, polyunsaturate, evangelisation, lancome
BalPrec	**athletics**, athlete, **tennis**, **football**, **rugby**, **baseball**, hobby, **racing**, specialty, **boxing**, **cricket**, entertainment, tournament, venue, **golf**
WeightedCosine	game, **football**, club, news, **tennis**, race, **racing**, **rugby**, **cricket**, **golf**, event, television, art, championship, **baseball**

weapon (noun)	
Cosine	**gun**, **missile**, bomb, weaponry, force, item, vehicle, warhead, tool, device, goods, instrument, equipment, measure, material
DiceGen2	**missile**, **gun**, bomb, device, tool, vehicle, item, instrument, goods, equipment, ingredient, threat, troops, measure, remedy
BalAPInc	weaponry, warhead, mycotoxin, valuable, gadget, **missile**, armament, yardstick, deterrent, safeguard, precaution, binoculars, power-up, explosive, breadwinner
ClarkeDE	medicinal, oakbrook, uim, chouilly, yore, introjection, endocytosis, glander, saurischian, amniocentesis, transsexualism, tidesman, mordor, guppies, sidmouth
BalPrec	**missile**, weaponry, safeguard, precaution, warhead, sanction, deterrent, ingredient, remedy, yardstick, gadget, artefact, valuable, bomb, aeroplane
WeightedCosine	**gun**, **missile**, bomb, weaponry, warhead, force, device, vehicle, tool, item, equipment, threat, **sword**, goods, instrument

support (noun)	
Cosine	assistance, commitment, help, resource, service, interest, benefit, control, **protection**, demand, strength, involvement, policy, responsibility, **approval**
DiceGen2	commitment, resource, assistance, interest, help, service, benefit, control, application, demand, strength, contribution, capability, opposition, influence
BalAPInc	assistance, **funding**, encouragement, co-operation, **backing**, **approval**, impetus, cooperation, help, **endorsement**, involvement, subsidy, leverage, **foothold**, participation
ClarkeDE	medicinal, thuringia, mordor, oakbrook, caulerpa, chouilly, egress, lebensraum, self-worth, biosynthesis, outworking, orkneys, buchlyvie, age-, yore
BalPrec	assistance, **funding**, commitment, involvement, help, **protection**, recognition, co-operation, resource, **approval**, expertise, encouragement, subsidy, benefit, responsibility
WeightedCosine	assistance, commitment, help, service, interest, resource, control, benefit, **backing**, application, demand, strength, **protection**, **approval**, policy

linguist (noun)

Cosine	sociologist, linguistics, archaeologist, anthropologist, researcher, psychologist, marxists, mathematician, scholar, commentator, educationist, taxonomist, historian, theorist, scientist
DiceGen2	sociologist, linguistics, archaeologist, anthropologist, mathematician, marxists, biologist, botanist, psychologist, physicist, commentator, pianist, proponent, theorist, educationist
BalAPInc	aglietta, demographer, taxonomist, tumin, liberal-historian, counterlife, hydrogeologist, vaill, gerontology, sexologist, centenarian, qalys, grave-goods, mafart, bailee
ClarkeDE	medicinal, ying, g.d, endocytosis, battison, saurischian, chouilly, zelan, macbryde, blacksell, ibaez, to-ing, wissenland, aggi, oakbrook
BalPrec	aglietta, taxonomist, educationist, grave-goods, sociologist, tumin, occultist, bailee, demographer, attitudinist, linguistics, vaill, pharmacologist, embryologist, bacteriologist
WeightedCosine	linguistics, sociologist, archaeologist, anthropologist, psychologist, mathematician, researcher, scholar, marxists, historian, commentator, theorist, educationist, **grammarian**, hydrogeologist

fabric (noun)

Cosine	**cloth**, yarn, curtain, garment, carpet, **material**, jacket, skirt, dress, stitch, pattern, rug, sweater, thread, finish
DiceGen2	**cloth**, yarn, curtain, garment, carpet, jacket, skirt, sweater, finish, dress, rug, trousers, thread, layer, stitch
BalAPInc	wall-hanging, bollworm, yarn, bedspread, outworking, buzz-word, **cloth**, **jacquard**, garment, coat-hanger, microfibre, counterpane, cladding, **swatch**, **sheeting**
ClarkeDE	medicinal, chouilly, orkneys, thuringia, mordor, introjection, wissenland, bollworm, oakbrook, m42, sidmouth, yore, outworking, ecms, aggi
BalPrec	yarn, garment, **cloth**, curtain, rug, carpet, stripe, stitch, tights, furnishings, **canvas**, skirt, sweater, thread, decor
WeightedCosine	**cloth**, yarn, curtain, garment, carpet, jacket, **material**, sweater, skirt, stitch, dress, trousers, **lace**, finish, thread

vegetable (noun)	
Cosine	fruit, salad, **potato**, food, **lettuce**, herb, ingredient, meal, meat, dish, prawn, garlic, fish, **cabbage**, **tomato**
DiceGen2	salad, fruit, **potato**, **lettuce**, herb, ingredient, meat, **mushroom**, **tomato**, garlic, prawn, **onion**, **cabbage**, dish, steak
BalAPInc	**veg**, caulerpa, worm-cake, cryptocorynes, roughage, parmesan, caviare, lectin, crudit, owenites, **gherkin**, rennet, **endive**, d.i.y, mollies
ClarkeDE	medicinal, zelan, macbryde, blacksell, ibaez, to-ing, aggi, caulerpa, acquisitiveness, chouilly, oakbrook, polyunsaturate, thuringia, sidmouth, guppies
BalPrec	**veg**, **lettuce**, foodstuff, roughage, time-cue, herb, worm-cake, salad, **cabbage**, spice, prawn, **spinach**, parmesan, produce, ingredient
WeightedCosine	fruit, salad, **lettuce**, **potato**, food, herb, prawn, meat, ingredient, meal, garlic, **tomato**, dish, **onion**, apple

fruit (noun)	
Cosine	**apple**, vegetable, tomato, **grape**, meat, food, potato, **pear**, herb, **melon**, leaf, **strawberry**, mushroom, **plum**, cheese
DiceGen2	**apple**, tomato, vegetable, **grape**, meat, potato, cheese, herb, leaf, mushroom, chocolate, food, **strawberry**, onion, **orange**
BalAPInc	chillus, **pear**, **melon**, **grape**, **kumquat**, vegetable, **plum**, rhubarb, yam, tomato, **citrus**, daphnium, **raisin**, **grapefruit**, **strawberry**
ClarkeDE	medicinal, aggi, zelan, macbryde, blacksell, ibaez, to-ing, polyunsaturate, thuringia, oakbrook, caulerpa, chouilly, nissel, ying, orkneys
BalPrec	vegetable, **grape**, **melon**, herb, **pear**, tomato, **apple**, **banana**, **strawberry**, potato, **plum**, mushroom, meat, chillus, chocolate
WeightedCosine	**apple**, tomato, **grape**, vegetable, meat, potato, **pear**, food, **melon**, leaf, **strawberry**, herb, mushroom, **plum**, cheese

treatment (noun)

Cosine	**therapy**, measure, care, procedure, use, method, remedy, practice, action, assessment, approach, investigation, change, condition, **intervention**
DiceGen2	**therapy**, measure, procedure, method, care, use, remedy, approach, assessment, technique, investigation, action, practice, solution, condition
BalAPInc	**therapy**, **medication**, remedy, **chemotherapy**, antibiotic, disclosure, **intervention**, investigation, evaluation, **reconsideration**, cure, assessment, referral, modification, monitoring
ClarkeDE	medicinal, mordor, chouilly, thuringia, oakbrook, egress, introjection, christianization, perdition, marksmanship, criterion-referencing, spiritualism, yore, g.d, orkneys
BalPrec	**therapy**, remedy, **intervention**, investigation, assessment, evaluation, **consideration**, measure, **medication**, dose, care, modification, improvement, procedure, disclosure
WeightedCosine	**therapy**, care, procedure, measure, method, use, action, remedy, approach, practice, investigation, assessment, change, programme, condition

travel (verb)

Cosine	**move**, **walk**, **go**, **work**, **pass**, **run**, **return**, live, **drive**, **fly**, arrive, **carry**, stay, **take**, **come**
DiceGen2	**walk**, **fly**, arrive, **move**, attend, **drive**, **ride**, stay, **pass**, **cross**, **wander**, **return**, **visit**, **head**, enter
BalAPInc	**journey**, **trudge**, **stray**, **tramp**, queue, urinate, **stroll**, **wander**, **taxi**, co-operate, **paddle**, **sail**, venture, clamber, **ski**
ClarkeDE	-drop, jive, foal, **traipse**, busk, urinate, loaf, **taxi**, defecate, doss, hassle, fund-raise, bunk, liaise, empathize
BalPrec	**wander**, behave, **sail**, queue, **progress**, participate, venture, **march**, **swim**, **stray**, book, **retreat**, dance, communicate, **stroll**
WeightedCosine	**move**, **walk**, **pass**, **work**, **drive**, **fly**, **go**, **return**, live, **run**, arrive, **carry**, stay, talk, **visit**

take (verb)

Cosine	**make**, give, **go**, **use**, put, **bring**, come, find, see, **get**, **hold**, leave, provide, offer, **show**
DiceGen2	give, **make**, **use**, **go**, **bring**, find, put, come, **hold**, see, **get**, leave, offer, call, **show**
BalAPInc	put, **carry**, **bring**, **hold**, change, move, offer, **receive**, decide, allow, start, work, **draw**, **set**, **return**
ClarkeDE	jive, over-indulge, doss, disbar, urinate, comply, traipse, foal, action, defecate, card, accede, liaise, empathize, masturbate
BalPrec	put, **bring**, **hold**, **carry**, allow, offer, **go**, move, **set**, help, give, **require**, begin, **need**, provide
WeightedCosine	**make**, give, **go**, **use**, **get**, find, come, put, **bring**, see, **hold**, leave, **show**, provide, offer

distribute (verb)

Cosine	integrate, develop, **allocate**, provide, market, operate, produce, implement, expand, support, **supply**, obtain, sell, base, print
DiceGen2	integrate, market, **allocate**, print, implement, expand, **supply**, generate, ship, deliver, operate, **circulate**, purchase, handle, store
BalAPInc	reseller, **redistribute**, unbundle, resell, computerise, channel, **export**, process, interface, ship, **circulate**, **redeploy**, recompile, source, network
ClarkeDE	unbundle, videoconference, recompile, reseller, action, **reallocate**, deprave, multiprocess, reroute, coldwater, reallocate, vend, busk, jive, discomfort
BalPrec	**allocate**, **export**, process, integrate, **circulate**, ship, channel, implement, **redistribute**, **disperse**, **deploy**, computerise, market, invest, trade
WeightedCosine	integrate, develop, market, base, provide, operate, expand, **allocate**, sell, **supply**, support, print, produce, **circulate**, introduce

guarantee (verb)

Cosine	**ensure**, require, provide, limit, improve, reduce, achieve, restrict, pay, increase, allow, offer, affect, accept, maintain
DiceGen2	**ensure**, restrict, limit, grant, improve, **secure**, permit, enhance, earn, achieve, reduce, afford, define, assess, affect
BalAPInc	refund, accrue, backdate, jeopardize, deduct, **stipulate**, exempt, infringe, allocate, confer, repay, **insure**, imperil, compromise, recalculate
ClarkeDE	disbar, backdate, no-win, recalculate, fand, overcompensate, preselect, mishear, jive, turbocharge, clown, action, imperil, hunger, closet
BalPrec	allocate, assess, confer, repay, accrue, amount, restrict, preclude, compromise, benefit, attain, deduct, **insure**, exempt, dictate
WeightedCosine	**ensure**, limit, require, provide, reduce, improve, achieve, pay, grant, increase, offer, restrict, allow, **secure**, affect

address (verb)

Cosine	**discuss**, concern, **deal**, consider, **examine**, **present**, identify, regard, relate, tackle, understand, focus, reflect, **recognise**, refer
DiceGen2	**discuss**, **deal**, **examine**, tackle, concern, **present**, identify, confront, **direct**, relate, focus, investigate, inform, explore, pursue
BalAPInc	debate, action, clarify, analyze, evaluate, substantiate, comply, redress, familiarize, communicate, tackle, explicate, conceptualize, articulate, assimilate
ClarkeDE	action, jive, re-express, explicate, not, familiarize, foal, schoon, empathize, demean, lame, hospitalize, misspell, slight, redress
BalPrec	debate, evaluate, tackle, clarify, communicate, resolve, solve, implement, fulfil, focus, comply, confront, initiate, articulate, criticize
WeightedCosine	**discuss**, concern, **deal**, consider, **present**, **examine**, regard, relate, identify, refer, tackle, express, **direct**, raise, reflect

meet (verb)

Cosine	take, **see**, tell, make, ask, find, give, bring, know, work, put, help, hold, come, go
DiceGen2	tell, ask, bring, join, help, work, hold, send, **receive**, **face**, put, set, **visit**, allow, pay
BalAPInc	**fulfil**, interview, attend, benefit, **satisfy**, cope, invite, engage, persuade, implement, comply, compete, **contact**, assist, deal
ClarkeDE	empathize, preselect, foal, -drop, liaise, microprogram, martin, over-indulge, economise, fand, oversleep, intercede, jive, re-offend, best
BalPrec	**satisfy**, attend, invite, **fulfil**, benefit, deal, act, persuade, recognise, engage, agree, discuss, treat, travel, address
WeightedCosine	tell, take, **see**, ask, make, give, find, bring, know, work, hold, put, come, go, help

display (verb)

Cosine	show, reveal, develop, **produce**, reflect, demonstrate, represent, **exhibit**, require, contain, express, present, place, change, provide
DiceGen2	**exhibit**, reveal, demonstrate, reflect, possess, express, store, acquire, represent, develop, present, lack, combine, illustrate, retain
BalAPInc	**exhibit**, store, reproduce, communicate, convey, interface, conform, utilize, stock, evaluate, capitalise, manifest, optimize, utilise, classify
ClarkeDE	actualize, vandalize, empathize, foal, actualise, discomfort, hillwalk, jive, excerpt, child-rear, turbocharge, socrate, represent, optimize, serialise
BalPrec	**exhibit**, store, communicate, reproduce, convey, assess, evaluate, possess, function, appreciate, evolve, classify, exploit, interpret, view
WeightedCosine	show, develop, reveal, **produce**, reflect, demonstrate, represent, contain, require, **exhibit**, express, place, provide, present, describe

sneak (verb)

Cosine	tiptoe, busk, clamber, stroll, bluff, taxi, crawl, slink, dart, nip, swan, bump, tramp, rope, wander
DiceGen2	tiptoe, clamber, stroll, dart, amble, nip, peep, tramp, scamper, crawl, sprint, rope, bluff, bump, grope
BalAPInc	busk, queer, tiptoe, taxi, waylay, chug, traipse, swagger, swan, slink, standin', tip-toe, bluff, plop, scoot
ClarkeDE	busk, queer, coddle, chicken, traipse, tip-toe, bunk, not, swagger, economise, re-invest, clown, deprave, swan, skive
BalPrec	busk, tiptoe, taxi, queer, waylay, bluff, chug, swan, traipse, rope, swagger, rough, barge, standin', whack
WeightedCosine	tiptoe, clamber, stroll, slink, dart, crawl, bluff, **creep**, taxi, sprint, wander, **slip**, swan, tramp, bump

confirm (verb)

Cosine	indicate, reveal, suggest, accept, **demonstrate**, state, **explain**, **show**, report, reflect, express, consider, concern, agree, announce
DiceGen2	indicate, **demonstrate**, reveal, state, accept, report, **explain**, announce, suggest, express, acknowledge, reflect, deny, assume, imply
BalAPInc	**substantiate**, **corroborate**, **contradict**, **verify**, reiterate, disclose, underline, summarize, infer, **disprove**, endorse, dispel, **ascertain**, reconsider, **refute**
ClarkeDE	not, action, **corroborate**, socrate, fand, re-affirm, preselect, **substantiate**, mishear, vandalize, demob, deprave, countermand, explicate, even
BalPrec	disclose, **demonstrate**, acknowledge, underline, endorse, state, **contradict**, interpret, review, emphasise, indicate, infer, question, submit, **document**
WeightedCosine	indicate, reveal, suggest, accept, **demonstrate**, report, **explain**, state, **show**, express, announce, agree, concern, reflect, consider

Appendix C: Similarity measures for entailment detection

We present here the full set of results for entailment detection. Word-level similarity measures are in columns, fragment-level similarity measures are in rows. The result of the best combination is marked in bold.

	Cosine	**Lin**	**JaccardGen**	**DiceGen**	**DiceGen2**
Cosine	73.21	74.40	72.42	72.74	73.11
Lin	72.72	72.67	72.12	72.32	70.69
JaccardGen	72.64	74.34	71.13	71.73	74.20
DiceGen	73.59	74.74	72.25	72.64	74.08
DiceGen2	71.63	72.92	70.13	70.58	70.66
ClarkeDE	77.45	79.17	76.39	76.83	79.06
WeedsPrec	75.82	75.00	75.77	75.77	72.76
WeedsRec	66.52	66.20	66.13	65.85	62.65
WeedsF	72.93	72.62	72.34	72.51	70.85
AP	71.71	72.97	70.40	70.87	71.42
APInc	73.89	75.00	72.72	73.09	74.90
BalAPInc	74.54	75.76	73.59	73.87	76.24
LinD	74.37	73.54	74.07	74.08	70.92
BalPrec	72.84	71.70	72.67	72.34	71.38
WeightedCosine	72.79	73.68	72.23	72.34	71.58

	ClarkeDE	**WeedsPrec**	**WeedsRec**	**WeedsF**	**AP**
Cosine	73.31	75.25	73.62	74.35	71.90
Lin	72.84	73.54	71.64	72.52	71.33
JaccardGen	72.35	75.15	73.75	74.25	70.73
DiceGen	73.24	75.66	73.91	74.72	71.51
DiceGen2	71.17	73.69	72.13	72.90	69.46
ClarkeDE	77.28	79.94	78.37	79.10	75.93
WeedsPrec	75.97	75.89	73.96	74.95	75.20
WeedsRec	66.54	67.00	64.97	66.13	64.88
WeedsF	72.86	73.65	71.64	72.52	71.12
AP	71.42	73.86	72.13	72.87	69.72
APInc	73.62	75.78	74.16	75.00	72.26
BalAPInc	74.33	76.60	75.03	75.81	72.94
LinD	74.04	74.02	72.53	73.49	73.83
BalPrec	72.87	73.24	70.80	71.66	72.04
WeightedCosine	72.99	74.56	72.68	73.73	71.67

	APInc	**BalAPInc**	**LinD**	**BalPrec**	**W-Cosine**
Cosine	74.19	74.40	75.43	74.79	73.33
Lin	71.72	72.83	73.94	72.69	72.73
JaccardGen	73.30	73.66	75.72	74.72	72.82
DiceGen	74.30	74.66	75.89	75.15	73.70
DiceGen2	71.21	71.99	73.98	73.23	71.83
ClarkeDE	78.70	78.91	**80.08**	79.55	77.63
WeedsPrec	75.14	75.21	75.36	75.38	75.77
WeedsRec	64.94	65.83	66.84	66.49	66.39
WeedsF	71.91	72.75	73.94	72.54	72.87
AP	72.16	72.54	73.99	73.39	71.90
APInc	74.18	74.86	76.00	75.24	74.09
BalAPInc	75.71	75.52	76.75	76.11	74.77
LinD	72.65	73.48	73.42	73.81	74.33
BalPrec	72.03	72.75	73.76	72.84	72.73
WeightedCosine	72.96	73.44	74.32	74.11	72.80

References

Agichtein, E. & Gravano, L. (2000). Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries*, 85–94, ACM New York, NY, USA. 11

Alfonseca, E. & Manandhar, S. (2002). Extending a lexical ontology by a combination of distributional semantics signatures. *EKAW '02 Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web.*, 1–7. 61

Andersen, Ø., Briscoe, T., Buttery, P., Carroll, J., Medlock, B., Parish, T. & Watson, R. (2012). Text Processing Tools and Services from iLexIR Ltd. Tech. rep. 18

Andersen, Ø. E., Nioche, J., Briscoe, E. J. & Carroll, J. (2008). The BNC parsed with RASP4UIMA. In *Proceedings of the Sixth International Language Resources and Evaluation Conference (LREC08)*, Marrakech, Morocco. 27

Androutsopoulos, I. & Malakasiotis, P. (2010). A survey of paraphrasing and textual entailment methods. *Journal of Artificial Intelligence Research*, **38**, 135–187. 96

Apostolova, E., Tomuro, N. & Demner-Fushman, D. (2011). Automatic extraction of lexicosyntactic patterns for detection of negation and speculation scopes. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers*, 283–287. 37

Arora, S., Mayfield, E., Penstein-Rosé, C. & Nyberg, E. (2010). Sentiment Classification using Automatically Extracted Subgraph Features. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*. 118

Bar-Haim, R., Dagan, I., Dolan, B., Ferro, L., Giampiccolo, D., Magnini, B. & Szpektor, I. (2006). The Second PASCAL Recognising Textual Entailment Challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, 1–9, Citeseer. 94, 100

Baroni, M. & Lenci, A. (2011). How we BLESSed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, Edinburgh. 63

Bentivogli, L., Cabrio, E., Dagan, I., Giampiccolo, D., Leggio, M. L. & Magnini, B. (2010). Building Textual Entailment Specialized Data Sets: a Methodology for Isolating Linguistic Phenomena Relevant to Inference. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. 97

Berant, J., Dagan, I. & Goldberger, J. (2010). Global learning of focused entailment graphs. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, Section 6, 1220–1229, Association for Computational Linguistics. 88, 96

Biemann, C. (2005). Ontology learning from text: A survey of methods. *LDV Forum*, **20**, 75–93. 61

Bisson, G., Nédellec, C. & Cañamero, D. (2000). Designing clustering methods for ontology building-The Mo'K workbench. In *ECAI Ontology Learning Workshop*. 61

Breiman, L. (2001). Random forests. *Machine learning*, 5–32. 10

Briscoe, T. & Carroll, J. (2006). Evaluating the accuracy of an unlexicalized statistical parser on the PARC DepBank. In *Proceedings of the COLING/ACL*

on Main conference poster sessions, July, 41–48, Association for Computational Linguistics, Morristown, NJ, USA. 19, 29, 117, 127

Briscoe, T., Carroll, J. & Watson, R. (2006). The second release of the RASP system. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, July, 77–80, Association for Computational Linguistics, Sydney, Australia. 18, 29, 32, 51, 79, 81, 98, 103, 125, 133

Burnard, L. (2007). Reference Guide for the British National Corpus (XML Edition). Tech. rep. 27

Cahill, A., Burke, M., O'Donovan, R., Riezler, S., van Genabith, J. & Way, A. (2008). Wide-Coverage Deep Statistical Parsing Using Automatic Dependency Structure Annotation. *Computational Linguistics*, **34**, 81–124. 125

Caraballo, S. A. (1999). Automatic construction of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, 120–126. 61

Carreras, X. (2007). Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, vol. 7, 957–961. 118

Carroll, J., Briscoe, T. & Sanfilippo, A. (1998). Parser evaluation: a survey and a new proposal. In *Proceedings of the 1st International Conference on Language Resources and Evaluation*. 32

Chapelle, O., Scholkopf, B. & Zien, A. (2006). *Semi-Supervised Learning*. 12

Charniak, E. & Johnson, M. (2005). Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics - ACL '05*, **1**, 173–180. 117

Charniak, E., Blaheta, D., Ge, N., Hall, K., Hale, J. & Johnson, M. (2000). BLLIP 1987-89 WSJ Corpus Release 1. Tech. rep. 27

Chen, J., Ji, D., Tan, C. L. & Niu, Z. (2006). Relation extraction using label propagation based semi-supervised learning. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL - ACL '06*, July, 129–136, Association for Computational Linguistics, Morristown, NJ, USA. 11

Cimiano, P. & Staab, S. (2005). Learning concept hierarchies from text with a guided hierarchical clustering algorithm. In *ICML-Workshop on Learning and Extending Lexical Ontologies by using Machine Learning Methods*. 61

Clark, S. & Curran, J. R. (2004). Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. 20

Clark, S. & Curran, J. R. (2007). Formalism-independent parser evaluation with CCG and DepBank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, vol. 45, 248–255. 127, 129, 133

Clarke, D. (2009). Context-theoretic semantics for natural language: an overview. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, March, 112–119, Association for Computational Linguistics. 70, 103

Cohen, P. R. (1995). *Empirical Methods for Artificial Intelligence*. The MIT Press, Cambridge, MA. 86, 125

Collins, M. (2000). Discriminative reranking for natural language parsing. In *The 17th International Conference on Machine Learning (ICML)*. 117

Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine learning*. 10, 20, 21, 87

Curran, J. R. (2003). *From distributional to semantic similarity*. Ph.D. thesis, University of Edinburgh. 64, 67, 103

Curran, J. R., Clark, S. & Bos, J. (2007). Linguistically motivated large-scale NLP with C&C and Boxer. *Proceedings of the ACL 2007 Demonstrations Session (ACL-07 demo)*, 33–36. 20, 98

Dagan, I., Lee, L. & Pereira, F. (1997). Similarity-based methods for word sense disambiguation. *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, 56–63. 73

Dagan, I., Lee, L. & Pereira, F. C. N. (1999). Similarity-based models of word cooccurrence probabilities. *Machine Learning*, **31**, 1–31. 59

Dagan, I., Glickman, O. & Magnini, B. (2006). The PASCAL Recognising Textual Entailment Challenge. *Machine Learning Challenges*, 177–190. 95

Dice, L. R. (1945). Measures of the amount of ecologic association between species. *Ecology*, **26**, 297–302. 66, 103

Dolan, W. & Brockett, C. (2005). Automatically constructing a corpus of sentential paraphrases. *Proc. of IWP*, 9–16. 106

Farkas, R., Vincze, V., Móra, G., Csirik, J. & Szarvas, G. (2010). The CoNLL-2010 Shared Task: Learning to Detect Hedges and their Scope in Natural Language Text. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning — Shared Task*, 1–12, Association for Computational Linguistics. 33, 35, 36, 48, 49, 105

Francis, W. N. & Kučera, H. (1979). Brown corpus manual. Tech. rep. 28

Georgescul, M. (2010). A Hedgehop over a Max-Margin Framework Using Hedge Cues. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning—Shared Task*, July, 26–31. 37

Gildea, D. (2001). Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, 167–202. 17, 20, 114

Grefenstette, G. (1994). *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers, Norwell, MA, USA. 67

Haghighi, A. D., Ng, A. Y. & Manning, C. D. (2005). Robust textual inference via graph matching. In *Proceedings of the conference on Human Language*

Technology and Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Morristown, NJ, USA. 57, 95, 104

Hara, T., Miyao, Y. & Tsujii, J. (2007). Evaluating the Impact of Retraining a Lexical Disambiguation Model on Domain Adaptation of an HPSG Parser. In *Proceedings of the 10th Conference on Parsing Technologies*, June, 11–22. 32

Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics (COLING '92)*, July, 539, Association for Computational Linguistics, Morristown, NJ, USA. 61

Hickl, A., Bensley, J., Williams, J., Roberts, K., Rink, B. & Shi, Y. (2006). Recognizing textual entailment with LCC's GROUNDHOG system. In *Proceedings of the Second PASCAL Challenges Workshop*. 95

Hockenmaier, J. (2003). *Data and models for statistical parsing with Combinatory Categorial Grammar*. Ph.D. thesis. 20

Inui, K., Sornlertlamvanich, V., Tanaka, H. & Tokunaga, T. (1997). A new formalization of probabilistic GLR parsing. In *Proceedings of the 5th International Workshop on Parsing Technologies*. 19

Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 547–579. 66

Joachims, T. (1998). Text categorization with support vector machines: learning with many relevant features. In *10th European Conference on Machine Learning (ECML-98)*, 137–142, Springer, Chemnitz, Germany. 10

Joachims, T. (1999). Making large scale SVM learning practical. Tech. rep. 21, 87

Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, **32**. 11

Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika*, **30**, 81–93. 68

Kilicoglu, H. & Bergler, S. (2008). Recognizing speculative language in biomedical research articles: a linguistically motivated perspective. *BMC bioinformatics*, **9 Suppl 11**, S10. 37

Kim, J. D., Ohta, T., Tateisi, Y. & Tsujii, J. (2003). GENIA corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics*, **19**, 180–182. 32, 130

King, T. H., Crouch, R., Riezler, S., Dalrymple, M. & Kaplan, R. M. (2003). The PARC 700 dependency bank. In *Proceedings of the EACL03: 4th International Workshop on Linguistically Interpreted Corpora (LINC-03)*, 1–8. 29, 127

Klein, D. & Manning, C. D. (2004). Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, 478, Association for Computational Linguistics. 17, 115

Kohonen, T. (1990). The Self-Organizing Map. *Proceedings of the IEEE*. 11

Kotlerman, L., Dagan, I., Szpektor, I. & Zhitomirsky-Geffet, M. (2010). Directional distributional similarity for lexical inference. *Natural Language Engineering*, **16**, 359–389. 63, 70, 71, 75, 82, 96

Kruskal, W. H. (1958). Ordinal measures of association. *Journal of the American Statistical Association*, **53**, 814–861. 68

Kübler, S., McDonald, R. & Nivre, J. (2009). Dependency Parsing. In *Synthesis Lectures on Human Language Technologies*, vol. 2, 1–127. 98

Kullback, S. (1959). *Information Theory and Statistics*. John Wiley and Sons, New York. 72

Kullback, S. & Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, **22**, 79–86. 72

Lafferty, J., McCallum, A. & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 282–289, Citeseer. 10, 22, 42

Lee, L. (1999). Measures of distributional similarity. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, 25–32, Association for Computational Linguistics. 74

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady*, vol. 10, 707–710. 101

Light, M., Qiu, X. Y. & Srinivasan, P. (2004). The language of bioscience: Facts, speculations, and statements in between. *Proceedings of BioLink 2004 Workshop on Linking Biological Literature, Ontologies and Databases: Tools for Users*, 17–24. 36

Lin, D. (1993). Principle-based parsing without overgeneration. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics (ACL '93)*, 112–120. 78

Lin, D. (1998). Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational linguistics-Volume 2*, 768–774, Association for Computational Linguistics. 69, 71, 104

Lin, D. & Pantel, P. (2001). Discovery of inference rules for question-answering. *Natural Language Engineering*, **7**, 343–360. 96

Lintean, M. C. & Rus, V. (2009). Paraphrase Identification Using Weighted Dependencies and Word Semantics. In *Proceedings of the FLAIRS-22*, vol. 22, 19–28. 95

Lloyd, S. P. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, **28**, 129–137. 11

MacCartney, B. & Manning, C. D. (2008). Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd Inter-*

national Conference on Computational Linguistics, 521–528, Association for Computational Linguistics. 96

Manning, C. D. & Schütze, H. (1999). *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA. 66, 67, 73

Marcus, M., Kim, G., Marcinkiewicz, M. A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K. & Schasberger, B. (1994). The Penn Treebank: annotating predicate argument structure. In *ARPA Human Language Technology Workshop*, 114–119. 29

Marcus, M. P., Santorini, B. & Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 1–22. 29, 127

Maslow, A. (1968). *Toward a Psychology of Being*. 13

Maxwell III, J. T. & Kaplan, R. M. (1993). The interface between phrasal and functional constraints. *Computational Linguistics*. 29

McClosky, D., Charniak, E. & Johnson, M. (2006). Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, June, 152–159, Association for Computational Linguistics, Morristown, NJ, USA. 117

McDonald, R. & Nivre, J. (2011). Analyzing and integrating dependency parsers. *Computational Linguistics*. 12

Medlock, B. (2008). Exploring hedge identification in biomedical literature. *Journal of Biomedical Informatics*, **41**, 636–54. 37

Medlock, B. & Briscoe, T. (2007). Weakly supervised learning for hedge classification in scientific literature. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, vol. 45, 992–999, Prague, Czech Republic. 33, 36, 37

Miller, G. A. (1995). WordNet: a lexical database for English. *Communications of the ACM*, **38**, 39–41. 30, 79

Morante, R. (2009). Descriptive analysis of negation cues in biomedical texts. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC10)*, 1429–1436. 105

Morante, R. & Daelemans, W. (2009). Learning the scope of hedge cues in biomedical texts. In *Proceedings of the Workshop on BioNLP*, June, 28–36, Association for Computational Linguistics. 37, 46

Morante, R., Asch, V. V. & Daelemans, W. (2010). Memory-based resolution of in-sentence scopes of hedge cues. In *CoNLL-2010: Shared Task*, July, 40. 37, 55, 56

Ng, D., Honnibal, M. & Curran, J. R. (2010). Reranking a wide-coverage CCG parser. In *Proceedings of the Australasian Language Technology Association Workshop 2010*, 90. 117

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S. & Marsi, E. (2007). MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, **13**, 1. 98

Noreen, E. W. (1989). *Computer Intensive Methods for Testing Hypotheses: An Introduction*. Wiley, New York. 86, 125, 126

Øvrelid, L., Velldal, E. & Oepen, S. (2010). Syntactic scope resolution in uncertainty analysis. In *Proceedings of the 23rd international conference on computational linguistics*, August, 1379–1387. 37

Özgür, A. & Radev, D. R. (2009). Detecting speculations and their scopes in scientific text. *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, 1398–1407. 37

Paaß, G., Kindermann, J. & Leopold, E. (2004). Learning prototype ontologies by hierachical latent semantic analysis. 61

Pearson, K. (1895). Note on regression and inheritance in the case of two parents. *Royal Society Proceedings*, **58**, 241. 64

Pestian, J. P., Brew, C., Matykiewicz, P., Hovermale, D. J., Johnson, N., Cohen, K. B. & Duch, W. (2007). A shared task involving multi-label classification of clinical free text. In *Proceedings of the Workshop on BioNLP 2007 Biological, Translational, and Clinical Language Processing (BioNLP '07)*, Association for Computational Linguistics, Morristown, NJ, USA. 33

Petrov, S., Barrett, L., Thibaux, R. & Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL (ACL '06)*, 433–440, Association for Computational Linguistics, Morristown, NJ, USA. 17, 115

Petrov, S., Das, D. & McDonald, R. (2011). A universal part-of-speech tagset. *ArXiv:1104.2086*. 81

Plank, B. & van Noord, G. (2008). Exploring an auxiliary distribution based approach to domain adaptation of a syntactic disambiguation model. In *Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, 9–16, Association for Computational Linguistics, Manchester, UK. 117, 121

Poon, H. & Domingos, P. (2009). Unsupervised Semantic Parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 61

Poon, H. & Domingos, P. (2010). Unsupervised ontology induction from text. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL '10)*. 61

Read, J., Velldal, E., Oepen, S. & Øvrelid, L. (2011). Resolving Speculation and Negation Scope in Biomedical Articles with a Syntactic Constituent Ranker. In *Proceedings of the Fourth International Symposium on Languages in Biology and Medicine (LBM 2011)*. 38

Rei, M. & Briscoe, T. (2010). Combining manual rules and supervised learning for hedge cue and scope detection. In *Proceedings of the 14th Conference on Natural Language Learning*, July, 56. 15

Rei, M. & Briscoe, T. (2011). Unsupervised Entailment Detection between Dependency Graph Fragments. In *Proceedings of the 2011 Workshop on Biomedical Natural Language Processing, ACL-HLT 2011*, 10. 17, 72

Rei, M. & Briscoe, T. (2013). Parser lexicalisation through self-learning. In *In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013)*. 17

Rodgers, J. L. & Nicewander, W. A. (1988). Thirteen ways to look at the correlation coefficient. *American Statistician*, **42**, 59–66. 64

Sammons, M., Vydiswaran, V. V. & Roth, D. (2010). "Ask not what textual entailment can do for you...". In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 1199–1208, Association for Computational Linguistics. 97

Sampson, G. R. (1995). *English for the Computer: The SUSANNE Corpus and Analytic Scheme*. Clarendon Press (Oxford University Press). 28

Sekine, S. (1997). The domain dependence of parsing. In *Proceedings of the fifth conference on Applied natural language processing*, vol. 1, 96–102, Association for Computational Linguistics, Morristown, NJ, USA. 17, 114

Smucker, M. D., Allan, J. & Carterette, B. (2007). A comparison of statistical significance tests for information retrieval evaluation. *Proceedings of the sixteenth ACM conference on information and knowledge management (CIKM '07)*, 623. 86

Snow, R., Jurafsky, D. & Ng, A. Y. (2005). Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems*. 61, 96

Snow, R., Jurafsky, D. & Ng, A. Y. (2006). Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL (ACL '06)*, July, 801–808, Association for Computational Linguistics, Morristown, NJ, USA. 61

Spearman, C. (1904). The proof and measurement of association between two things. *The American journal of psychology*, **15**, 72–101. 65

Specht, D. F. (1990). Probabilistic neural networks. *Neural networks*, **3**. 10

Szarvas, G. (2008). Hedge classification in biomedical texts with a weakly supervised selection of keywords. In *Proceedings of 46th Meeting of the Association for Computational Linguistics*, June, 281–289. 37

Szpektor, I. & Dagan, I. (2008). Learning entailment rules for unary templates. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING '08)*, 849–856, Association for Computational Linguistics, Morristown, NJ, USA. 72, 96

Szpektor, I., Tanev, H., Dagan, I. & Coppola, B. (2004). Scaling web-based acquisition of entailment relations. In *Proceedings of EMNLP*, vol. 4, 41–48. 96

Tang, B., Wang, X., Wang, X., Yuan, B. & Fan, S. (2010). A Cascade Method for Detecting Hedges and their Scope in Natural Language Text. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning - Shared Task*, July, 13–17. 37

Tateisi, Y., Miyao, Y., Sagae, K. & Tsujii, J. (2008). GENIA-GR: a Grammatical Relation Corpus for Parser Evaluation in the Biomedical Domain. In *Proceedings of LREC*, 1942–1948. 32, 130

Taylor, A., Marcus, M. & Santorini, B. (2003). The Penn treebank: an overview. *Treebanks*. 29

Thelen, M. & Riloff, E. (2002). A bootstrapping method for learning semantic lexicons using extraction pattern contexts. *Proceedings of the Conference on Empirical methods in Natural Language Processing - (EMNLP '02)*, 214–221. 11

Tomita, M. (1987). An efficient augmented-context-free parsing algorithm. *Computational Linguistics*, **13**. 19

Ushioda, A. (1996). Hierarchical clustering of words and application to NLP tasks. In *Proceedings of the Fourth Workshop on Very Large Corpora*, 28–41. 61

van Noord, G. (2007). Using self-trained bilexical preferences to improve disambiguation accuracy. In *Proceedings of the 10th International Conference on Parsing Technologies*, June, 1–10, Association for Computational Linguistics, Morristown, NJ, USA. 116, 118, 121

Vapnik, V. (1982). *Estimation of dependences based on empirical data*. Springer-Verlag, New York. 20, 87

Velldal, E., Øvrelid, L. & Oepen, S. (2010). Resolving Speculation: MaxEnt Cue Classification and Dependency-Based Scope Rules. In *CoNLL-2010: Shared Task*, July, 48. 37

Velldal, E., Øvrelid, L., Read, J. & Oepen, S. (2012). Speculation and negation: Rules, rankers, and the role of syntax. *Computational Linguistics*. 38

Vincze, V., Szarvas, G., Farkas, R., Móra, G. & Csirik, J. (2008). The BioScope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC Bioinformatics*, **9 Suppl 11**, S9. 15, 32, 34, 37, 48

Vlachos, A. (2010). Semi-supervised learning for biomedical information extraction. Tech. Rep. 791. 12

Vlachos, A. & Craven, M. (2010). Detecting Speculative Language Using Syntactic Dependencies and Logistic Regression. In *Proceedings of the Four-*

teenth Conference on Computational Natural Language Learning - Shared Task, July, 18–25. 37

Wagner, A. (2000). Enriching a lexical semantic net with selectional preferences by means of statistical corpus analysis. In *ECAI Workshop on Ontology Learning*. 61

Wallach, H. M. (2004). Conditional random fields: An introduction. Tech. rep. 22

Watson, R. (2006). RASP Evaluation Schemes. Tech. rep. 18

Watson, R. (2007). *Optimising the speed and accuracy of a Statistical GLR Parser*. Ph.D. thesis. 18

Watson, R., Carroll, J. & Briscoe, T. (2005). Efficient extraction of grammatical relations. In *Proceedings of 9th Int. Workshop on Parsing Technologies (IWPT05)*, October, 160–170, Association for Computational Linguistics, Morristown, NJ, USA. 18

Watson, R., Briscoe, T. & Carroll, J. (2007). Semi-supervised training of a statistical parser from unlabeled partially-bracketed data. *Proceedings of the 10th International Conference on Parsing Technologies - IWPT '07*, 23–32. 19, 127, 129

Weeds, J. & Weir, D. (2005). Co-occurrence retrieval: A flexible framework for lexical distributional similarity. *Computational Linguistics*. 59

Weeds, J., Weir, D. & McCarthy, D. (2004). Characterising measures of lexical distributional similarity. *Proceedings of the 20th international conference on Computational Linguistics - COLING '04*. 69, 70

Weeds, J., Weir, D. & Keller, B. (2005). The distributional similarity of sub-parses. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, 7–12, Association for Computational Linguistics, Morristown, NJ, USA. 102, 106

Weeds, J. E. (2003). *Measures and applications of lexical distributional similarity*. Ph.D. thesis, University of Sussex. 73

Witschel, H. F. (2005). Using decision trees and text mining techniques for extending taxonomies. In *Proceedings of the Workshop on Learning and Extending Lexical Ontologies by using Machine Learning*. 61

Witten, I. H., Moffat, A. & Bell, T. C. (1999). *Managing Gigabytes. Compressing and Indexing Documents and Images*. Academic Press, San Diago, CA USA, 2nd edn. 64

Yeh, A. (2000). More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th conference on Computational Linguistics*, vol. 2, 947, Association for Computational Linguistics, Morristown, NJ, USA. 125

Zhitomirsky-Geffet, M. & Dagan, I. (2009). Bootstrapping Distributional Feature Vector Quality. *Computational Linguistics*, **35**, 435–461. 62, 78, 79, 82

Zhou, G., Zhao, J., Liu, K. & Cai, L. (2011). Exploiting Web-Derived Selectional Preference to Improve Statistical Dependency Parsing. In *49th Annual Meeting of the Association for Computational Linguistics*, 1556–1565. 117

Zhu, X. (2005). Semi-supervised learning literature survey. Tech. rep. 12

Zhu, X. & Ghahramani, Z. (2002). Learning from labeled and unlabeled data with label propagation. Tech. rep. 11