

Integrating Pattern-based and Distributional Similarity Methods for Lexical Entailment Acquisition

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

by

Shachar Mirkin

Under the supervision of

Dr. Ido Dagan, Department of Computer Science, Bar-Ilan University
Dr. Ari Rappoport, School of Computer Science and Engineering, the Hebrew
University of Jerusalem

School of Computer Science and Engineering
The Hebrew University of Jerusalem

December 2006

Acknowledgments

This thesis was completed thanks to the help of quite a few people, and I'm happy for the opportunity to thank them all.

First and foremost, I would like to thank my supervisor, Dr. Ido Dagan, who has invested a lot of his time in my work, who has supported me all along the way, has provided me with great ideas and insights and who has been a great person to learn from and work with in general.

I wish to thank my Hebrew University supervisor, Dr. Ari Rappoport for his backing and support for this work, for opening the door for me to exciting fields of NLP, and for giving me the opportunity to work on the things I find most interesting.

I deeply thank Maayan Geffet, with whom I have been working in the past couple of years, who has involved me in her work in my first NLP steps and has contributed a great deal to my own work.

I wish to thank Google, for providing me with an increased quota for web queries, making this work feasible.

Thanks to my colleagues at the Bar-Ilan NLP lab, Idan Szpektor, Roy Bar-Haim and Oren Glickman for their useful feedback in different stages of this work.

Thanks to my friend, Ido Omer, for various kinds of university-related help during the whole period of my Master's degree.

Deep thanks to my parents who've always helped and supported me, never expecting any gratitude.

Lastly, I would like to thank Inbal Ravid, Ofir and Gil, my beloved family, for their endless support, patience and love during the whole time I was working on this thesis.

Abstract

This thesis investigates algorithms for acquiring lexical semantic relationships, applied to the lexical entailment relation. Our main contribution is a novel conceptual integration between the two prior distinct acquisition paradigms for lexical relations – the pattern-based and the distributional similarity approaches. The integrated method exploits mutual complementary information of the two approaches to obtain candidate relations and informative characterizing features. Then, a small size training set is used to construct a more accurate supervised classifier, showing significant increase in both recall and precision over the original approaches.

Table of Contents

Acknowledgments.....	2
Abstract.....	3
Table of Contents.....	4
1 Introduction.....	5
2 Background.....	7
2.1 Acquiring Lexical Semantic Relationships.....	7
2.1.2 Existing Resources of Semantic Relationships.....	10
2.2 Lexical Acquisition Paradigms.....	11
2.2.1 The Pattern-based Approach.....	11
2.2.1.1 Automatic Learning of Patterns.....	14
2.2.1.2 Using the Web as a Corpus.....	14
2.2.2 The Distributional Similarity Approach.....	15
2.2.3 Our Motivation: The Complementary Nature of the Approaches.....	16
2.2.4 Earlier Combined Methods.....	18
3 An Integrated Approach for Lexical Entailment Acquisition.....	20
3.1 Pattern-based Extraction Module.....	20
3.2 Candidate Acquisition.....	22
3.2.1 Pattern-based Candidates.....	22
3.2.2 Distributional Similarity Candidates.....	23
3.3 Feature Construction.....	24
3.4 Training and Classification.....	27
4 Empirical Results.....	29
4.1 Data Set and Annotation.....	29
4.2 Results.....	29
4.3 Analysis.....	32
5 Conclusion and Future Work.....	38
References.....	40
Appendix A: Web Pattern-based Extraction Module.....	44
Appendix B: POS Tagger and NP Chunker Reference.....	49
Appendix C: Patterns Grammar.....	51
Appendix D: SVM Training Procedure.....	53

1 Introduction

Learning lexical semantic relationships is a fundamental task needed for most text understanding applications. Several types of lexical semantic relations were proposed as a goal for automatic acquisition. These include acquisition of lexical ontological relations such as synonymy, hyponymy and meronymy, aiming to automate the construction of WordNet-style relations (Fellbaum, 1998). Another common target is learning general distributional similarity between words, following Harris' Distributional Hypothesis (Harris, 1968). Recently, an applied notion of entailment between lexical items (Geffet, 2006) was proposed as capturing major inference needs which cut across multiple semantic relationship types (see Section 2 for further background).

The literature suggests two major approaches for learning lexical semantic relations: distributional similarity and pattern-based. The first approach recognizes that two words (or multi-word terms), e.g. *country* and *state* are semantically similar based on distributional similarity of the different contexts in which the two words occur. However, the distributional method (e.g. Lin, 1998) identifies a somewhat loose notion of semantic similarity, such as between *country* and *city*, which does not ensure that the meaning of one word can be substituted by the other. The second approach is based on identifying joint occurrences of the two words within particular patterns, which typically directly indicate concrete semantic relationships. For example, using the pattern NP_1 and other NP_2 (where NP stands for Noun Phrase), a hyponymy relationship between *copper* and *goods* can be identified from the sentence: “*The Egyptians came hither from the land of the Blacks bringing gold, which they exchange for copper and other goods*”. The pattern-based approach (e.g. Sundblad, 2002) tends to yield more accurate hyponymy and (some) meronymy relations, but is less suited to acquire synonyms which only rarely co-occur within short patterns in texts. It should be noted that the pattern-based approach is commonly applied also for information and knowledge extraction to acquire factual instances of concrete meaning relationships (e.g. *born in, located at*) rather than generic lexical semantic relationships in the language.

While the two acquisition approaches are largely complementary, there have been just few attempts to combine them. In this thesis we propose a methodology for integrating distributional similarity with the pattern-based approach. In particular, we focus on learning the lexical entailment relationship between common nouns and noun phrases (to be distinguished from learning relationships for proper nouns which usually falls within the knowledge acquisition paradigm).

The underlying idea is to first identify candidate relationships by both the distributional approach, which is applied exhaustively to a local corpus, and the pattern-based approach, applied to the web. Next, each candidate is represented by a unified set of distributional and pattern-based features. Finally, using a small training set we devise a supervised (SVM) model that classifies the candidate relations as correct or incorrect.

To implement the integrated approach we developed state of the art pattern-based acquisition methods and utilized a distributional similarity method that was previously shown to provide superior performance for lexical entailment acquisition. Our empirical results show that the integrated method significantly outperforms each approach in isolation, as well as the naïve combination of their outputs. Overall, our method reveals complementary types of information that can be obtained from the two approaches.

A preliminary version of this thesis was reported in (Mirkin et al., 2006).

2 Background

This section describes the task addressed in this work, the motivation and related work done in the field. The idea of automatic lexical semantic acquisition is presented (2.1), then, *lexical entailment* is defined within the framework of *textual entailment* and the motivation for using it is explained (2.1.1). Next, the use of existing repositories of lexical relationships is shortly reviewed, highlighting the need to perform the task automatically (2.1.2). Then, the two prominent paradigms for carrying out the task are described (2.2.1, 2.2.2), as well as the pros and cons of each one. Finally, some previous attempts to combine the two paradigms are reviewed (2.2.3).

2.1 Acquiring Lexical Semantic Relationships

Automatic acquisition of lexical semantic relationships is a well studied task whose importance is two-fold. First, it helps understanding the language structure in order to improve its modeling, and second, it is useful for various Natural Language Processing (NLP) applications. For instance, identifying that two words are synonyms can be useful for applications using term expansion such as Question Answering (QA) and Information Retrieval (IR); finding a label term for a class to which a given term belongs can be useful for Text Categorization; and finding relationships between verbs can help determine the order of a sequence of events. Existing repositories of semantic relationships exist and are being widely used, but as they are incomplete and suffer from some inherent deficiencies (see 2.1.2), automatic acquisition becomes necessary as well.

Various relation types have been studied, among them some are generic relations between words such as synonymy or meronymy (part-whole) and some are more specific, such as the relation between a book and its author or between a movie and its stars, which are more related to Information Extraction (IE) and Knowledge Acquisition (KA). In addition, it is common to search for terms which adhere to a *similarity* relation, usually through methods based on term distribution. In this work we present a lexical acquisition method that is applied for the *substitutable lexical entailment* relation.

2.1.1 Substitutable Lexical Entailment

It was recently noted by Geffet and Dagan (2004, 2005) that distributional similarity captures a quite loose notion of semantic similarity, as exemplified by the typical pair *country – party* which was identified by Lin's similarity measure (Lin, 1998; See Section 2.2.2). Consequently, they proposed a definition for the *substitutable lexical entailment* (*lexical entailment* for brevity) relation, which conforms to the general framework of applied *textual entailment* (Dagan et al., 2005). Under this framework, a text expression “*T*” (*Text*) entails another text expression “*H*” (*Hypothesis*) if the meaning of *H* can be inferred from the meaning of *T*, as would be judged by most people. For example, the Text “*Cavern Club sessions paid the Beatles £15 evenings and £5 lunchtime.*” is entailing the Hypothesis “*The Beatles perform at Cavern Club at lunchtime.*” as the meaning (or “truth”) of the second expression can be inferred from the first.

Textual Entailment may be considered as the applied version of logical entailment, where an event or state is a logical consequence of another, e.g. *snoring* entails *sleeping* (Moldovan et al., 2004). This relation is useful for various NLP applications and especially for precision-oriented ones. Question Answering, Information Retrieval, Information Extraction, Multi-text Summarization, Paraphrasing in Text Generation, Text Categorization, Anaphora Resolution and Machine Translation (MT) evaluation are all applications for which the textual entailment is useful, as exemplified hereunder. In a QA task, textual entailment is useful to identify texts that entail the hypothesized answer (Harabagui and Hickl, 2006). For instance, given the question “*Which team won the world cup in 2006?*”, the text “*Italy’s victory at the 2006 FIFA World Cup*” entails the hypothesized answer “*Italy won the 2006 world cup*”. A QA system that is capable of textual entailment inference can identify texts that entail the answer even if their structure is different from the question’s. In multi-document summarization, textual entailment enables identifying redundant sentences and excluding them from the summary (Dagan et al., 2005). IE applications (e.g. Romano et al., 2006) can benefit from recognizing that two text fragments express the same relation, and extract the relation even if it does not appear within a predefined extraction pattern because it appears in a text entailing this pattern. In IR, textual entailment recognition is useful to identify that the document’s text entails the query (the hypothesis). For example, if the query is “*Delhi attack*”, documents containing the term “*Delhi bomb blast*” can also be retrieved, even if they do not explicitly contain the word “*attack*” (Geffet, 2006).

Within the textual entailment framework, lexical substitution is the task of replacing specific terms in the text with ones that will maintain the meaning of the complete context, and lexical entailment is the relation that must hold for lexical substitution to be applicable. A large acquisition effort of lexical entailment relationships is therefore required if one is to build a textual entailment engine. Lexical substitution has been previously used mainly in the form of term expansion using resources such as WordNet for the purpose of increasing recall in NLP applications (Moldovan and Mihalcea, 2000; Negri, 2004). These works, though, were done in application-dependent manner, with neither a standard definition nor a standard methodology determining which relations should be used for the substitution task.

It was suggested that lexical entailment captures major application needs in modeling lexical variability, generalized over several types of known ontological relationships. For example, in Question Answering, the word *company* in a question can be substituted in the text by *firm* (synonym), *automaker* (hyponym) or *subsidiary* (meronym), all of which entail *company*. Typically, hyponyms entail their hypernyms and synonyms entail each other, while entailment holds for meronymy only in certain cases.

The ***Substitutable Lexical Entailment*** relation is defined in detail in (Geffet, 2006). It is a directional relation between two terms w , v where w entails v under a common sense of the terms if the following two conditions are satisfied:

1. ***Word meaning entailment***: The meaning of w implies a meaning of v . To assess this condition Geffet suggests an operational test through transformation of the terms to existential statements. For example, for the terms *company* (w) and *organization* (v), the following must be satisfied: If a *company* exists (in some world), the existence of *organization* must be implied. This pair satisfies the condition under the sense of *company* as *commercial organization*, but the pair *organization* - *company* fails to comply, as the existence of an *organization* does not necessarily imply the existence of *company*. We conclude, then, that *organization* does not entail *company*. This condition is not enough on its own for determining substitutable lexical entailment, since pairs such as *car* - *producer* may also satisfy it, while not maintaining the entire text meaning if being substituted, as required by the second condition.
2. ***Semantic Substitutability***: w can semantically substitute v in some natural contexts, such that the meaning of the modified context textually-entails the meaning of the

original one. This condition is directly derived from the motivation for substitutable lexical entailment, but does not suffice on its own since some term pairs may be substitutable in certain contexts, but not in general. *demonstrator* can be substituted by *worker* in a text on workers' strikes and demonstrations, but this entailment does not hold in general. The issue of context-specific entailment is further discussed in our analysis (Section 4).

2.1.2 Existing Resources of Semantic Relationships

Large repositories of lexical relationships already exist, the prominent of them being WordNet. This is a manually-constructed ontology, inspired by psycholinguistic theories about human lexical memory which contains over 150,000 nouns, verbs, adjectives and adverbs organized in *synsets* - sets of synonyms - each representing a lexical concept in the language. Over 200,000 relationships are defined to date between synsets in WordNet. Some of the relations included in WordNet are synonymy, hyponymy-hypernymy (*a kind of*) and meronymy (*part of*) for nouns, and synonymy, troponymy-hypernymy (*one way to*) for verbs.

WordNet has grown to become an almost indispensable tool in the domain of ontological relations. As each of its entries is manually edited, it is characterized by a high level of accuracy. Hence, it is frequently used as the reference for evaluation of lexical acquisition methods (Hearst, 1992; Pantel et al., 2004; Pantel and Ravichandran, 2004; Davidov and Rappoport, 2006) and as a repository of established lexical relationships (Girju et al., 2003). Additionally, it is commonly used as a knowledgebase for term expansion or word labeling, applied to various purposes like Information Retrieval (Voorhees, 1994), Question Answering (Pasca and Harabagiu, 2001; Hovy et al., 2002), Text Categorization (Rosso et al., 2004) and Word Sense Disambiguation (Kwong, 2001). However, WordNet, as other manually constructed ontologies, has its drawbacks. Chklovski and Pantel (2004) and Curran (2005) point out three problems from which such ontologies suffer, that are directly related to the manual nature of their construction: (i) inconsistency in classification (ii) bias in coverage – some concepts are better covered than others (iii) limited coverage, especially for infrequent or domain-specific words. WordNet includes only correct usage cases and does not include plausible but not certain relationships, which are often useful for applications. In addition, WordNet does not include all relations required for NLP purposes, lexical entailment being one of them. These issues emphasize the need for automatic acquisition and ontology-construction methods to coexist with the manually constructed repositories.

2.2 Lexical Acquisition Paradigms

The literature includes two main approaches for automatic acquisition of lexical semantic relations - the pattern-based approach and the distributional similarity approach. In most cases these methods were used in isolation, and even when combined it was done in pipeline architecture, where the pattern-based method receives the output of the distributional similarity method for the purpose of classification or labeling. In the following sections we describe the approaches, highlighting the advantages and disadvantages of each, especially in comparison to each other. This background lays the ground for the idea of a tight integration between the two approaches, which might achieve better results by benefiting from their complementary characteristics.

2.2.1 The Pattern-based Approach

This section presents the pattern-based approach, describing several works using patterns for the purpose of lexical acquisition. Then, some methods for automatic learning of patterns are described and finally, the utilization of the web for pattern-based lexical relationship acquisition is shortly discussed.

The general idea of the pattern-based approach is to search for co-occurrences of terms in language patterns that are known to indicate the relationship of interest. Hearst (1992) pioneered the use of lexico-syntactic patterns for automatic extraction of lexical semantic relationships. She acquired hyponymy relations from a corpus based on bootstrapping an initial small predefined set of highly indicative patterns, such as “*X, . . . , Y and/or other Z*”, and “*Z such as X, . . . and/or Y*”, where *X* and *Y* are extracted as hyponyms of *Z*.

Hearst’s Algorithm includes the following steps:

- 1) Start with a set of seeds - known pairs of the relation of interest.
- 2) Search the corpus for instances where the pairs occur near each other.
- 3) Create new patterns for the relation from the found occurrences.
- 4) Search the corpus for instances of the new patterns.
- 5) Extract terms from the instances found and use the new terms to search the corpus again (return to step 2).

1	NP_1 such as NP_2
2	Such NP_1 as NP_2
3	NP_1 or other NP_2
4	NP_1 and other NP_2
5	NP_1 ADV known as NP_2
6	NP_1 especially NP_2
7	NP_1 like NP_2
8	NP_1 including NP_2
9	NP_1 -sg is (a OR an) NP_2 -sg
10	NP_1 -sg (a OR an) NP_2 -sg
11	NP_1 -pl are NP_2 -pl

Table 1: The patterns we used for entailment acquisition based on (Hearst, 1992) and (Pantel et al., 2004). Capitalized terms indicate variables. *pl* and *sg* stand for plural and singular forms.

Starting with three manually defined patterns, Hearst identified through this process three more patterns which occur frequently enough and normally indicate hyponymy. Patterns 1-4, 6 and 8 in Table 1 are these six patterns. The results were evaluated with WordNet and of 106 pairs where both words existed in this repository, 61 (57.5%) hyponymy relationships were found in it.

Hearst work was preliminary from several aspects, mostly mentioned by Hearst herself. First, the pattern creation step of the algorithm is not well defined as the task of creating patterns from a list of occurrences is not trivial. Hearst handled this task manually, while automatic methods were later developed (see Section 2.2.1.1 for details). Second, Hearst dealt mostly with occurrences where the words occur as unmodified nouns, thus avoiding handling complex noun phrases. Our own system is designed to work at the NP-level and learn multi-word terms as well, as explained in Section 3 and Appendix A. Finally, Hearst considered all extracted relationships as correct ones, applying no further selection techniques (possibly due to the small number of relationships available in the corpus). This was later extended by works using statistical measures to rank pairs or to set a threshold for pair selection as described hereunder. To perform more systematic selection Etzioni et al. (2004) applied a supervised Machine Learning algorithm (Naïve Bayes), using pattern statistics as features. Their work was done within the IE framework, aiming to extract semantic relation instances for proper nouns which occur quite frequently in indicative patterns. In our work we incorporate and extend the supervised learning step for the more difficult task of acquiring general language relationships between common nouns. To deal with the low number of relationships found in corpora, recent works extracted relationships from the web using queries to find pattern instances (2.2.1.2). We do so as well, as detailed in Section 3.

Pattern	Example
whole NN[-PL]'s POS part NN[-PL]	...building's basement...
part NN[-PL] of PREP {the a} DET mods [JJ NN]* whole NN	...basement of a building...

Table 2: Patterns used by Berland and Charniak for meronym extraction. In this notation each part of the pattern is followed by its Part of Speech tag.

Following Hearst, similar techniques were applied to predict other relations and to fill the gaps mentioned above. Berland and Charniak (1999) applied a method based on Hearst's technique to extract part-whole relations (meronyms) from a corpus. Given a single word denoting an entity with recognizable parts, their system would find and rank words that denote parts - either physical or conceptual - of that entity. They used two manually constructed patterns (Table 2) and searched a corpus for occurrences of terms in the patterns. Then, using a statistical metric designed to combine pattern-word likelihood and number of occurrences, they ranked the extracted words. Berland and Charniak did not limit their search to unmodified nouns, and instead extracted the head from each noun phrase, a heuristic which worked for most cases, but which - by their own analysis - introduced errors to the results (e.g. *conditioner* instead of *air conditioner*). In Section 3.2.1 we explain how we dealt with this issue. 55% of their top 50 words and 70% of their top 20 words for each given seed word were judged as correct parts. It should be noted, though, that for their input they used as seeds only words of high probability to have parts (e.g. *building*) and not arbitrary words.

Further works attempted to acquire hyponymy and meronymy relationships using lexical or lexico-syntactic patterns (Sundblad, 2002; Pantel et al., 2004), and web page structure was exploited to extract hyponymy relationships by Shinzato and Torisawa (2004). Chklovski and Pantel (2004) used 35 patterns to extract a set of ontological relations between verbs, such as similarity, strength and antonymy. Synonyms, on the other hand, are rarely found in such patterns.

In addition to their use for learning generic lexical semantic relations, patterns were commonly used to learn instances of concrete semantic relations for IE and QA, such as *Author-Title*, *Person-Birthday*, *Person-Invention*, and *Film-Actor* to name a few (Brin, 1998; Ravichandran and Hovy, 2002; Etzioni et al, 2004).

2.2.1.1 Automatic Learning of Patterns

An additional component of the pattern-based paradigm is the automatic generation of patterns, a common approach being the application of bootstrapping to enrich the pattern set. Initially, some instances of the sought relation are found based on a set of manually defined patterns. Then, additional co-occurrences of the related terms are retrieved, from which, with the use of various techniques, common structures in the text are extracted and ranked. The highest ranking among them become new extraction patterns (Finkelstein and Morin, 1999; Yangarber et al., 2000). Ravichandran et al. (2002) automatically learned patterns from the web for Question Answering using suffix trees. Given an example from a required question type, like *birth-year*, and an example instance like *Mozart* (question term) and *1756* (answer term), they search for co-occurrences of the question and the answer terms in the same sentence. These sentences are passed through a suffix tree constructor which finds all substrings of all lengths of the sentence along with their counts – the number of times they were found. Only substrings containing both question and answer terms are kept and for the common ones the terms are replaced with variable slots for the question and answer, thus creating patterns. For the above example, patterns such as “*born in <ANSWER>, <NAME>*” and “*<NAME> (<ANSWER> -)*” are created. The process is repeated with several examples for each question type resulting in a list of patterns ranked by their precision – the ratio between the number of instances with the question and answer in the pattern and the number of instances with the question term alone. Pantel et al., (2004) suggested an algorithm based on edit distance for learning patterns at multiple levels, such as the lexical level and the Part of Speech (POS) level, and Davidov and Rappoport (2006) generated patterns through a completely unsupervised method combining high frequency words and content words.

The list of effective patterns found for ontological relations has pretty much converged in the literature. Amongst these, Table 1 lists the patterns that we found useful and utilized in our work. This set of patterns was chosen in order to extract hyponymy, meronymy and possibly synonymy (with the last 3 patterns) relationships, for the study of our integrated approach, but by no means attempts to be exhaustive.

2.2.1.2 Using the Web as a Corpus

One of the limitations that early works such as of Hearst and Berland and Charniak faced was the bounded scope of their corpora. Patterns identify rather specific and informative

structures within particular co-occurrences of the related words. Consequently, they are relatively reliable and tend to be more accurate than distributional evidence. On the other hand, they are susceptible to data sparseness in a limited size corpus. To obtain sufficient coverage, recent works such as (Chklovski and Pantel, 2004; Pantel et al., 2004) applied pattern-based approaches to the web. These methods form search engine queries that match likely pattern instances, which may be verified by post-processing the retrieved texts.

Using the web allows utilizing simpler extraction patterns and applying simpler processing techniques, as the terms are more likely to be found more than once (Etzioni et al., 2004). Additionally, the style of the web opens the door to learn relationships that may be absent from a more formally edited corpus. This, however, is also a drawback, as not all occurrences are syntactically coherent, inserting noise to applications using syntax processing tools such as POS taggers. A key limitation is the number of queries one can submit to a search engine within a time period, mainly due to legal or ethical bounds. Another issue is evaluating the results of a web-based learning system: Calculating recall is not straight-forward as the actual number of results is unknown and the numbers of found pages provided by search engines are inaccurate estimates. For that reason, *relative recall* may be used instead of absolute recall (Pantel et al., 2004). In this case, the success of the system in obtaining as many correct relationships possible is compared to the performance of another system. Szpektor et al. (2004) used another useful measure - the *yield* of the system, measuring in absolute numbers how many correct patterns were obtained from the web for each input relation.

2.2.2 The Distributional Similarity Approach

The general idea behind distributional similarity is that words which occur within similar contexts are semantically similar (Harris, 1968). In a computational framework, words are represented by feature vectors, where features are context words weighted by a function of their statistical association with the target word. The degree of similarity between two target words is then determined by a vector comparison function, such as cosine. Amongst the many proposals for distributional similarity measures, Lin98 (Lin, 1998) is maybe the most widely used one for NLP. Lin defines the similarity between two words w, v as follows:

$$sim_{Lin98}(w, v) = \frac{\sum_{f \in T(w) \cap T(v)} (I(w, f) + I(v, f))}{\sum_{f \in T(w)} I(w, f) + \sum_{f \in T(v)} I(v, f)}$$

where a feature f is a word w' related to w through some syntactic relation. For example, the word *cell* has a feature (*subj-of, absorb*) created from a sentence where *cell* acts as the subject of the verb *absorb*. $I(w, f)$ is an association measure between w and w' , like the Pointwise Mutual Information (PMI) between the two words. $T(w)$ denotes the set of active features of the word w , i.e. the features of w with association measure above a certain threshold. This measure computes the ratio between the sum of the PMI values of the common features of w and v and the sum of PMI values of each one alone. The idea is that the stronger the common features are, the stronger the measure value will be.

Geffet and Dagan (2004) proposed a new method (denoted here GD04) for computing similarity which employs Lin's similarity measure and assigns new feature weights through a bootstrapped feature weight function. For a pair of words w, v and a feature f , the feature's weight is defined as follows:

$$weight_{GD04}(w, f) = \sum_{v \in WS(f) \cap N(w)} sim(w, v)$$

where $sim(w, v)$ is some kind of initial similarity score between the words, such as Lin98; $WS(f)$ is the word set of the feature f , i.e. the words for which it is an active feature, and $N(w)$ denotes the semantic neighborhood of w - the set of words v for which $sim(w, v) > s$, s being a predefined threshold. That is, the feature score is the sum of similarity scores between w and each of the highly similar words which share this feature. The idea is to rank higher common features of similar words, which is suggested as a desirable behavior. Similarity score between the words is then re-computed using Lin98.

These and other measures are commonly used in methods exploiting distributional similarity for various tasks such as word sense discovery (Pantel and Lin, 2002), Text Generation (Lin, 2006) and Named Entity Recognition (Zhao, 2004), as well as for the task of semantic relationship acquisition (Pantel and Ravichandran, 2004; Lindén and Piitulainen, 2004). For the purpose of predicting the lexical entailment relation between noun terms, GD04 was used by Geffet and Dagan (2004, 2005), showing improved results compared to Lin98.

2.2.3 Our Motivation: The Complementary Nature of the Approaches

Each of the two approaches described above identifies different lexically entailing term pairs and - more importantly - different types of terms. This, by itself is a clear motivation for using both methods in conjunction as more entailment relationships may be discovered this

way. Additionally, each method suffers from inherent problems limiting the accuracy of its results, which can sometimes be improved with the help of complementary information extracted from the other method. We explain some of these problems here, illustrating why an integrated method can be helpful. A more complete examination of the problems and behavior of the methods is found in our empirical analysis in Section 4, based on the results of our own research.

Distributional similarity measures are typically computed through exhaustive processing of a corpus, and are therefore applicable to corpora of bounded size. Yet, a large enough corpus needs to be processed in order to achieve significant statistics, as shown in (Pantel and Ravichandran, 2004). As mentioned, distributional methods produce a rather loose notion of similarity between terms rather than indicating a concrete relation. Furthermore, distributional similarity is typically defined as a symmetric relation, making it insufficient by itself for predicting directional relationships like hyponymy, meronymy or lexical entailment. For that reason, determining the exact relation with this method is difficult without extra evidence (positive or negative) such as constraints or patterns.

A major problem of pattern-based methods is that patterns do not always uniquely identify a specific relation. Moldovan et al. (2004) showed that many lexico-syntactic patterns are ambiguous, i.e., more than one semantic relation can be expressed by the same pattern and the relation in use is often dependent on the context. This notion was also mentioned by Hearst (1992) referring to the limited success in extracting meronyms. Consider for example the pattern *NP including NPList*. This pattern may indicate hyponymy, as in “*The PowerShot G3 offers advanced camera features including a maximum shutter speed of 1/2000 second ...*” where *maximum shutter speed of 1/2000 second* is a hyponym of *advanced camera feature*. On the other hand, the same pattern can indicate meronymy in cases such as: “*The hotel agreed to remove barriers to access throughout the hotel, including the hotel entrance, parking, guest rooms, restaurant and restroom.*” in which *hotel entrance, parking, guest room, restaurant* and *restroom* are all meronyms of *hotel*. Patterns are subjected to errors due to context-dependent relationships or due to dependencies not captured within the pattern’s scope. Extra evidence is often needed to remove such ad-hoc errors, and the evidence may come from distributional similarity information. Furthermore, some relations rarely occur in patterns. Synonymy is a prominent example, as it rarely serves a purpose to use words with identical or nearly identical meaning within a short-distance co-occurrence. On the other hand, hyponymy or meronymy relations have many patterns in the literature identified for them.

Pattern	Example
from X to Y	from <i>adversary</i> to <i>ally</i>
either X or Y	either <i>ally</i> or <i>adversary</i>

Table 3: Patterns used by Lin et al. (2003) to distinguish antonyms from synonyms.

2.2.4 Earlier Combined Methods

The motivation for integrating the pattern-based and the distributional approaches is apparent. However, only few attempts - as described below - were made to use the two methods in conjunction, mostly to classify or to label distributional similarity output using lexico-syntactic patterns, in pipeline architecture. We aim to achieve tighter integration of the two approaches, as described in the Section 3.

(Caraballo, 1999) used patterns to label distributional similar words for the purpose of automatic construction of a hierarchy of nouns. Working with a parsed corpus, Caraballo first extracted words from two syntactic dependencies: conjunctions (e.g. *executive VP and treasurer*) and appositions (e.g. *James Rosenfield, a former CBS contractor*). The idea is that nouns fulfilling these dependencies tend to be semantically related. Using the stemmed head of each noun phrase as her pool of words, she created a feature vector for each noun, counting how many times each of the other words appeared in conjunction or apposition with it. Using cosine function, similarity between the vectors was calculated and bottom up clustering was applied: At each step the two most similar nouns were joined under a common parent node. To compute the similarity of the new node with other nodes, the weighted average similarity of each of its children with the other node was computed. The process was repeated until all nodes had one root. The next step in the tree construction was to assign labels for each internal node. To this end, patterns known to indicate hyponymy (patterns 3 & 4 in Table 1) were used to extract hyponymy relationships from the corpus. For each leaf a vector of hypernyms was constructed, counting the number of times each noun in the vector appeared in hypernymy relation with it. For each internal node a similar vector was constructed by joining the vectors of its children. The hypernym with the most occurrences was chosen as the node’s label. Through this algorithm Caraballo created a tree of 20,000 leaves with 650 internal nodes, with accuracy of 33% to 60%, depending of the strictness of the evaluation.

Lin et al. (2003) used lexical patterns to distinguish between synonyms and antonyms of distributionally similar pairs of nouns. To be able to tell the types apart, “Patterns of incompatibility” were used - patterns in which the occurrence of a pair of words is likely to indicate they are not synonyms. Lin used the two patterns in Table 3 and required that the ratio between the number of near occurrences of the pair and the number of its occurrences in the two patterns would be above an experimentally set threshold. While achieving very good results (86.4% precision and 95% recall in identifying synonyms), it should be noted that the task, as well as the data set, were very specific, especially given the fact that each pair of words chosen for evaluation was known to be either a pair of synonyms or of antonyms. In practice, aside from the fact that most nouns do not have antonyms, synonyms and antonyms constitute only a part of distributionally similar words, and other means are required to tell them apart from the rest (in this case a dictionary was used).

Pantel and Ravichandran (2004) used syntactic patterns for naming clusters of distributionally similar words, creating hyponymy relations between each member of a class and the class label. For each class a committee is created, i.e. a set of words that unambiguously represent the class. Each word in the committee is represented by a feature vector of syntactic relations with it, and by averaging the feature scores of words in the committee, a signature is created for each class. Next, each of a chosen set of syntactic relationships (like apposition and the *such as* pattern that we also used) is searched for in the signature and the scores of each term found in these relationships are summed. The term with the highest score is defined as the label of the class, i.e. the hypernym of each of its members.

3 An Integrated Approach for Lexical Entailment Acquisition

This section describes our integrated approach for acquiring lexical entailment relationships, applied to common nouns terms. The algorithm receives as input a *target term* (single or multi-word) and aims to automatically acquire a set of terms that entail or are entailed by it. We denote a pair consisting of the input target term and an acquired entailing/entailed term as *entailment pair*. Entailment pairs are directional, as in *bank* \rightarrow *company*. By this notation, a pair of synonyms corresponds to two entailment pairs, one for each entailment direction.

Our approach applies a supervised learning scheme, using SVM, to classify candidate entailment pairs as correct or incorrect. The SVM training phase is applied to a small constant number of training pairs, yielding a classification model that is then used to classify new test entailment pairs. The designated training set is also used to tune some additional parameters of the method. Overall, the method consists of the following main components:

- 1: Acquiring candidate entailment pairs for the input term by pattern-based and distributional similarity methods (Section 3.2);
- 2: Constructing a feature set for all candidates based on pattern-based and distributional information (Section 3.3);
- 3: Applying SVM training and classification to the candidate pairs (Section 3.4).

The first two components, of acquiring candidate pairs and collecting features for them, utilize a generic module for pattern-based extraction from the web, which is described first in Section 3.1.

3.1 Pattern-based Extraction Module

The general pattern-based extraction module receives as input a set of lexico-syntactic patterns (as in Table 1) and either a target term or a candidate pair of terms. It then searches the web for occurrences of the patterns with the input term(s). A small set of effective queries is created for each pattern-terms combination, aiming to retrieve as much relevant data with as few queries as possible.

Each pattern has two variable slots to be instantiated by candidate terms for the sought relation. Accordingly, the extraction module can be used in two modes: (a) receiving a single target term as input and searching for instantiations of the other variable to identify candidate related terms (as in Section 3.2); (b) receiving a candidate pair of terms for the relation and

searching pattern instances with both terms, in order to validate and collect information about the relationship between the terms (as in Section 3.3). Utilities like Google proximity search (Previously used by Chklovski and Pantel, 2004) provide a useful tool for these purposes, as it allows using a wildcard which might match either an un-instantiated term or optional words such as modifiers. For example, the query *"such ** as *** (war OR wars)"¹* is one of the queries created for the input pattern *such NP₁ as NP₂* and the input target term *war*, allowing new terms to match the first pattern variable. The automatically constructed queries, covering the possible combinations of multiple wildcards, are submitted to Google² and a specified number of snippets is downloaded, avoiding duplicates. The above query may retrieve snippets like *"As time went by the catalogue of these disasters expanded, progressing from such crises as earthquakes, floods, famines, wars, and the death of kings ..."* from which the candidate pair *war* → *crisis* can, at the end of the processes, be extracted. For the candidate entailment pair *war* → *crisis*, the first variable is instantiated as well. The corresponding query would be: *"such * (crisis OR crises) as *** (war OR wars)"*. This technique allows matching terms that are sub-parts of more complex noun phrases as well as multi-word terms, or that are not immediately adjacent to the literal (constant) strings of the pattern, such as in the case of lists, as can be seen in the example above.

The retrieved snippets are passed through a word splitter and a sentence segmenter³, while filtering individual sentences that do not contain all search terms (all snippets contain the terms, but snippets may include more than one sentence). Next, the sentences are processed with the OpenNLP⁴ Part Of Speech Tagger and NP Chunker, producing chunked sentences as the following example, which was produced for the pair *warplane* → *aircraft*: *[NP US/NNP warplanes/NNS] and/CC [NP other/JJ coalition/NN aircraft/NN] "/" [VP arrived/VBD] [PP at/IN] [NP the/DT scene/NN] and/CC [VP provided/VBD] [NP continuous/JJ close/JJ air/NN support/NN] "/"*. Finally, pattern-specific regular expressions over the chunked sentences are applied to verify that the instantiated pattern indeed occurs in the sentence, and to identify variable instantiations.

A more detailed description of the module is found at Appendix A.

¹ Note that the quotation marks are part of the query

² <http://www.google.com/apis/>

³ Available from the University of Illinois at Urbana-Champaign, <http://l2r.cs.uiuc.edu/~cogcomp/tools.php>

⁴ www.opennlp.sourceforge.net/

3.2 Candidate Acquisition

Given an input target term we first employ pattern-based extraction to acquire entailment pair candidates and then augment the candidate set with pairs obtained through distributional similarity.

3.2.1 Pattern-based Candidates

At the candidate acquisition phase pattern instances are searched with one input target term, looking for instantiations of the other pattern variable to become the candidate related term (the first querying mode described in Section 3.1). We construct two types of queries, in which the target term is either the first or second variable in the pattern, which usually corresponds to finding either entailing or entailed terms that instantiate the other variable.

In the candidate acquisition phase we utilized patterns 1-8 in Table 1, which we empirically found as most suitable for identifying directional lexical entailment pairs. Patterns 9-11 are not used at this stage as they produce too much noise when searched with only one instantiated variable. About 35 queries are created for each target term in each entailment direction for each of the 8 patterns. For every query, the first n snippets are downloaded (we used $n=50$). The downloaded snippets are processed as described in Section 3.1, and candidate related terms are extracted, yielding candidate entailment pairs with the input target term.

A common practice when learning lexical relations is to extract the head of the noun-phrase (Berland & Charniak, 1999; Caraballo, 1999). However, quite often the entailment relation holds between multi-word noun-phrases rather than merely between their heads. For example, *trade center* lexically entails *shopping complex*, while *center* does not necessarily entail *complex*. Some methods limit the search to patterns including only unmodified nouns (Hearst, 1992), but as we do wish to extract multi-word terms, this solution will not do. On the other hand, many complex multi-word noun phrases are too rare to make a statistically based decision about their relation with other terms. Hence, we apply the following two criteria to balance these constraints:

1. For the entailing term we extract only the complete noun-chunk which instantiate the pattern. For example: we extract *housing project* \rightarrow *complex*, but do not extract *project* as entailing *complex* since the head noun alone is often too general to entail the other term.

2. For the entailed term we extract both the complete noun-phrase and its head in order to create two separate candidate entailment pairs with the entailing term, which will be judged eventually according to their overall statistics.

As it turns out, a large portion of the extracted pairs constitute hyponymy relations where one term is a modified version of the other, like *low interest loan* \rightarrow *loan*. We consider these as trivial entailment relations and thus removed them from the candidate list, along with numerous pairs including proper nouns, following the goal of learning entailment relationships for distinct common nouns.

Finally, we filter out the candidate pairs whose frequency in the extracted patterns is less than a threshold, which was set empirically to 3. Using a lower threshold yielded poor precision on the training set, while a threshold of 4 decreased recall substantially with just little positive effect on precision.

3.2.2 Distributional Similarity Candidates

As mentioned in Section 2, we employ the distributional similarity measure of Geffet and Dagan (2004) which was found effective for extracting non-directional lexical entailment pairs. Using local corpus statistics, this algorithm produces for each target noun a scored list of up to a few hundred words with positive distributional similarity scores.

Next, we need to determine an optimal threshold for the similarity score, considering words above it as likely entailment candidates. To tune such a threshold we followed the original methodology used to evaluate *GD04*. First, the top- k ($k=40$) similarities of each training term are manually annotated by the lexical entailment criterion (see Section 4.1). Then, the similarity value which yields the maximal micro-averaged F1 score is selected as threshold, suggesting an optimal recall-precision tradeoff. The selected threshold is then used to filter the candidate similarity lists of the test words.

Finally, we remove all entailment pairs that already appear in the candidate set of the pattern-based approach, in either direction (recall that the distributional candidates are non-directional) under the empirically observed rationale that if a pair was found by the pattern-based method in one direction, but not the other, the pair in the direction which was not found is unlikely to be entailing.

Each of the remaining candidates generates two directional pairs which are added to the unified candidate set of the two approaches.

3.3 Feature Construction

The next step is to represent each candidate by a set of features, suitable for supervised classification. To this end we developed a novel feature set based on both pattern-based and distributional data, collecting information from both the corpus and the web.

To obtain pattern statistics for each pair, the second mode of the pattern-based extraction module is applied (see Section 3.1). As in this case both variables in the pattern are instantiated by the terms of the pair, we could use all eleven patterns in Table 1, creating a total of about 55 queries per pair and downloading up to $m=20$ snippets for each query (some queries may not return as many as 20 snippets). The downloaded snippets are processed as described in Section 3.1 to identify pattern matches and to obtain relevant statistics for feature scores.

As detailed below, the features were designed with the motivation to capture information from one method that would be complementary to the other. Following is the list of feature types computed for each candidate pair.

Conditional Pattern Probability

This type of feature is created for each of the 11 individual patterns and is similar to the kind of feature used by Etzioni et al. (2004). For each pair and each entailment pattern, the feature intends to indicate how likely the pattern is to appear as connecting the two words of the pair, in turn aiming to indicate its entailment likelihood. The feature value is the conditional probability of having the pattern matched in a sentence given that the pair of terms appears in that sentence:

$$score(f_i^{pat}) = \Pr(\text{pattern}_i^{w_1, w_2} | w_1, w_2) \cong \frac{count(\text{pattern}_i^{w_1, w_2})}{count(w_1, w_2)}$$

where $i = 1, \dots, p$, p being the number of patterns used ($p=11$ in our case). $\text{pattern}_i^{w_1, w_2}$ is the event of pattern i being instantiated by w_1 and w_2 in a sentence, conditioned on the event of the two words appearing in a sentence from amongst the set of unique sentences retrieved by the queries of the pattern. This conditional probability is estimated by the ratio of the number of sentences in which the pattern is instantiated with the pair ($count(\text{pattern}_i^{w_1, w_2})$) and the number of sentences in which the pair occurs ($count(w_1, w_2)$).

This feature yields normalized scores for pattern matches regardless of the number of snippets retrieved for the given pair. This normalization is important in order to bring to equal grounds candidate pairs identified through either the pattern-based or through the distributional similarity approach, since the latter pairs tend to occur less frequently in patterns.

We also attempted estimating the conditional probability based on web hit counts, i.e., the number of results received from the search engine. Based on the hit count and the number of pattern-pair instantiations in the snippet sample we processed, we estimated the number of instantiations we could find have we downloaded all possible snippets for the query. The conditioning event was estimated by the hit count of the pair as two separate words.

This feature score turned out to be less useful than the one shown above. This may be due to the fact that search result counts are not very accurate, since results are not uniform – the first ones are more accurate, and since the results refer to a pair not in a sentence, but in a document⁵.

Aggregated Conditional Pattern Probability

This single feature is the conditional probability that *any* of the patterns match in a retrieved sentence, given that the two terms appear in it. Some pairs are strongly related with specific patterns while others' occurrences are more equally balanced between several patterns. It may be so that the second kind provides better evidence for entailment prediction, yet its single feature scores are relatively low. This feature comes to compensate for that by looking at the overall occurrences and not only at the occurrences of each pattern separately. It is calculated like the previous feature, with counts aggregated over all patterns:

$$score(f^{agg-pat}) = \Pr(pattern_{any}^{w_1, w_2} | w_1, w_2) \cong \frac{\sum_{i=1}^P count(pattern_i^{w_1, w_2})}{count(w_1, w_2)}$$

where $pattern_{any}^{w_1, w_2}$ denotes the instantiations of w_1 and w_2 in any pattern.

Conditional List-Pattern Probability

This feature was designed to eliminate the typical non-entailing cases of co-hyponyms (words sharing the same hypernym), which consist of a significant portion of the candidate pairs and

⁵ Technically, there is no way to limit the search to sentence-level only.

are mostly acquired by the distributional similarity method. We check for pair occurrences in lists, using an appropriate list pattern, $NP_1 \{,NP\}^* (and|&|or) NP_2$, expecting that correct entailment pairs would not co-occur there. The list pattern is incorporated into the standard patterns we employ, by placing the two words of the pair as list members in one of the two variable positions of the pattern. For example, for the sixth pattern in Table 1 and a candidate pair *war – illness*, we create a corresponding list query: “* like (*illness OR illnesses*) * (*war OR wars*)”, which fetches sentences like: “...concerns along with other risks like *illness, crime, war ...*”. The motivation to search for lists near entailment patterns is triple. First, we assume that this is a likely location to find lists. Second, it largely eliminates cases of ambiguity expressed by the list pattern, since NP_1 and NP_2 is likely to indicate other syntactic dependencies besides lists, like conjunctions between clauses. Lastly, it helped avoid duplicates occurring by retrieving the same snippet, once from an entailment pattern query and once from a list pattern query. The probability estimate, calculated like the previous feature, is expected to be a negative feature for the learning model.

For example, the query “* like (*chairman OR chairmen*) (*director OR directors*)” is created for the pattern NP_1 like NP_2 and the terms *chairman* and *director*. It yields sentences such as “A combination of experience and expertise can lead to appointment at top level positions like *chairman, directors and managing director...*” where the two terms are co-hyponyms, thus serving as evidence for non-entailment.

Relation Direction Ratio

The value of this feature is the ratio between the overall number of pattern matches for the pair and the number of pattern matches for the reversed pair (a pair created with the same terms in the opposite entailment direction):

$$score(f^{dir}) = \frac{count(pattern_{any}^{w_1, w_2})}{count(pattern_{any}^{w_2, w_1})}$$

For example, the pair *tea - goods* got a score of 71 for this feature, strongly indicating that this is the only direction of entailment. In cases where the denominator equaled zero, i.e. when no pattern occurrences were found in the direction w_2-w_1 , the score was set to twice the nominator’s value, i.e. twice the number of occurrences in the w_1-w_2 direction. This heuristic is a tradeoff between the need to reward pairs which are strongly directional, and the

necessity to keep their feature scores between reasonable bounds to avoid making the feature scores of the rest of the terms negligible.

We found that this feature strongly correlates with entailment likelihood. Interestingly, it does not deteriorate performance for synonymous pairs.

Distributional Similarity Score

The *GD04* similarity score of the pair (explained in Section 2.2.2) was used as a feature, aiming to reduce errors generated by the pattern-based approach. Pairs for which at least one of the terms does not appear in the local corpus (and therefore no similarity score can be generated for them) received the average *GD04* score for all the training set pairs. We also attempted adding *Lin98* similarity scores, computed over the same local corpus, but they appeared to be redundant.

Intersection Feature

Naturally, pairs that were found separately by two methods are more likely to be correct entailment pairs. We reflect this by a binary feature which indicates that a candidate pair was identified by both the pattern-based and distributional methods. In Section 4.2.1 we describe another experiment we conducted in order to examine the option to classify pairs as correct based solely on this feature.

In summary, the above feature types utilize mutual complementary pattern-based and distributional information. Using cross validation over the training set, we verified that each feature makes marginal contribution to performance when added on top of the remaining features.

3.4 Training and Classification

We used SVM^{light} (Joachims, 1999), a state-of-the-art SVM classifier, for entailment pair classification. Our training set consisted of a sample of the candidates obtained by the two methods for the training target terms. The candidates, represented by feature vectors that include a score for each of the features, were fed to the SVM learning module whose goal is to find an optimal separating hyperplane between the two classes of the examples. Using 10-fold cross-validation over the training set, we obtained the SVM configuration that yields an optimal micro-averaged F1 score, i.e. the score obtained by calculating F1 once from

accumulated numbers of correct and incorrect pairs from all the cycles combined. Through this optimization we chose the RBF kernel function and obtained optimal values for the J , C and the RBF's *gamma* parameters. The resulting *training model* was used by the SVM classifier to classify the test set candidates, which were represented by feature vectors in the same fashion. The candidate test pairs classified as correct entailments constitute the output of our integrated method.

Appendix D includes more technical details of the training procedure. More details about the data set can be found in Sections 4.1 and 4.2 below.

Pattern-based	Distributional	Total
1186	1420	2350

Table 4: The numbers of distinct entailment pair candidates obtained for the test words by each of the methods, and when combined.

4 Empirical Results

In this section the data set and annotation scheme of our experiments are described (4.1), then, the results (4.2) and our analysis (4.3) are detailed.

4.1 Data Set and Annotation

We received and utilized the experimental data set from Geffet and Dagan (2004). The dataset includes the similarity lists calculated by *GD04* for a sample of 30 target (common) nouns, computed from an 18 million word subset of the Reuters corpus⁶. The 30 nouns were sampled from the corpus nouns whose frequency exceeded 500, and out of them, we randomly picked a small set of 10 terms for training, leaving the remaining 20 terms for testing. Then, the set of entailment pair candidates for all nouns was created by applying the thresholding method of Section 3.2.2 to the distributional similarity lists, and by extracting pattern-based candidates from the web as described in Section 3.2.1.

Gold standard annotations for entailment pairs were created by three judges, taking the majority vote. The judges were guided to annotate as “Correct” the pairs conforming to the lexical entailment definition described in Section 2.1.1 using whichever lexical resources they deem fit, such as WordNet, the web and dictionaries. The obtained Kappa values (varying between 0.7 and 0.8) correspond to *substantial agreement* between the annotators on the task.

4.2 Results

The exact numbers of candidate entailment pairs collected for the test terms are shown in Table 4. Adopting the measure defined by Szpektor et al. (2004), we define the *yield* of our system as the number of correct relationships obtained for each input term. On average, 120 candidate entailment pairs were acquired for each target term.

⁶ Reuters Corpus, Volume 1, English Language, 1996-08-20 to 1997-08-19.

Method	P	R	F
<i>Distributional Similarity</i>	0.33	0.53	0.40
<i>Pattern-based</i>	0.44	0.61	0.51
<i>Naïve Combination</i>	0.36	1.00	0.53
<i>Integrated Classification</i>	0.57	0.69	0.62

Table 5: Precision, Recall and F1 figures for the test words under each method.

These figures highlight the markedly complementary yield of the two acquisition approaches, where only about 10% of all candidates were identified by both methods. It may be pointed out that this terms divergence can be attributed to the fact that one method uses a local corpus and the other uses the web. Indeed, this is partly the case. However, within a training set sample of 700 candidate pairs, 84% of the distributional pairs were found in patterns through our web queries and 85% percent of the pattern-based candidates were assigned a distributional similarity score from the corpus. Apparently, although both corpora did include information about most of the candidate pairs of both sources, only a small amount of them were suggested by both methods, further emphasizing the complementary nature of the methods. Each method requires its own resource to acquire candidates from: Querying the web is a more efficient way to extract the relatively rare pattern-based relations, while distributional methods require exhaustive processing of an entire corpus. Obtaining all terms from a single resource, be it the web or a corpus, would be less efficient in terms of time, number of queries or corpus size. Using our methodology, we first discover terms in each method’s optimal setting and then use the other resource to find additional information about these terms.

The SVM classifier was trained on a quite small annotated sample of 700 candidate entailment pairs which were generated by the 10 training terms. Table 5 presents comparative results for the integrated classifier, for the two sets of candidates produced by each method alone without classification, and for the union of these two sets (referred as *Naïve Combination*). The results were computed for an annotated random sample of about 400 candidate entailment pairs for the test terms. Following common pooling evaluations in Information Retrieval, recall is calculated relatively to the total number of correct entailment pairs acquired by both methods together.

The first two rows of the table show quite moderate precision and recall for the candidates of each separate method. The next row shows the great impact of method combination on recall, relative to the amount of correct entailment pairs found by each method alone, validating the complementary output of the approaches. The integrated classifier, applied to the combined set of candidates, succeeds to increase precision substantially by 21 points (a relative increase of almost 60%), which is especially important for many precision-oriented applications like Information Retrieval and Question Answering. The precision increase comes with the expense of some recall, yet having F1 improved by 9 points. The integrated method yielded on average about 30 correct entailments per target term. Its classification *accuracy* (percent of correct classifications) reached 70%, nearly doubling the naïve combination's accuracy (36%⁷).

It is impossible to directly compare our results with those of other works on lexical semantic relationships acquisition, since the particular task definition and dataset are different. As a rough reference point, our result figures do match those of related papers reviewed in Section 2, while we notice that our setting is relatively more difficult since we excluded the easier cases of proper nouns and extracted multi-word terms. A later work of Geffet and Dagan (2005) addressed the same task as ours using an improved distributional method. Their method obtained higher precision but substantially lower recall as they considered only distributional candidates. Further research is suggested to investigate integrating their approach with ours.

4.2.1 Additional Experiments

In this section we present a couple more experiments we conducted in order to compare with the method described above, motivating the method which was eventually chosen.

4.2.1.1 Training Each Set Alone

An additional experiment we conducted was to separately train and classify the candidates of each of the two methods, and then unify the results. This was done in order to check the possibility that the candidates from different sources are of different nature and that a more homogenous set may produce a better classification model. This experiment achieved similar precision, but lower recall and F1 scores than the joint model (P=0.57, R=0.66, F=0.61).

⁷ In the naïve method all candidates are classified as “correct”, hence the accuracy equals the precision.

Pattern-based	Distributional
abduction → abuse	assault ↔ abuse
government → organization	government ↔ administration
drug therapy → treatment	budget deficit → gap
palm oil → goods	copper → goods
woodland → environment	sentence ↔ conviction

Table 6: Typical entailment pairs acquired by the integrated method, illustrating Section 4.3. The columns specify the original method that produced the candidate pair.

4.2.1.2 Assuming that Overlapping Pairs are Correct

Another option we considered was to classify all the pairs that were produced by the two methods as entailing and to run the SVM classifier only on the non-overlapping pairs. This experiment was motivated by the fact that overlapping pairs acquired for the training set terms reached precision of 62.5%, a somewhat higher value than that of the method we presented above. In this set, overlapping pairs consisted of about 6% of the candidates (of each method), but consisted of 11% of the entailing ones. This suggested that if the same recall/precision values would be achieved when training the classifier using all the candidates except the overlapping ones, we can increase recall by about 3% simply by assuming the overlapping ones are correct. That is as 11% of the entailing pairs would have a recall of 100% instead of 69%. Conducting this experiment we achieved the following results: For the non-overlapping pairs the SVM classifier achieved 55% and 65% of precision and recall respectively. When adding the overlapping candidates, classified as entailing, we got 56% precision, 70% recall and an F1 value of 62.5%, 0.3% higher than the result achieved in the final integrated method.

As none of the two above experiments showed a significant improvement of our main method, we decided to stick to the more straight forward method, as presented in Section 3.

4.3 Analysis

The integrated method reveals and highlights some interesting features of the behavior of each of the two approaches. Below are the details of our analysis of the results, followed by an error analysis, explaining the reasons for some of the errors and suggesting how our method can be improved in future work. Table 6 lists some typical examples for correct

Pattern-based	Distributional
government → corporation	training → programme
performance → music	opposition → government
long distance → bill	head of state → chairman
goods → time	goods → vehicle
privatisation → robbery	gap trade → deficit

Table 7: Typical erroneous candidate entailment pairs successfully removed by the integrated system.

entailment pairs extracted by the system, while Table 7 shows some typical errors successfully removed by it.

1) Discovering different relation types

Analysis of the data confirmed that the two methods tend to discover different types of relations. As expected, the distributional similarity method contributed most (75%) of the synonymous word pairs that were correctly classified as mutually entailing (e.g. *assault* ↔ *abuse* in Table 6). On the other hand, about 80% of all correctly identified hyponymy relations were produced by the pattern-based method (e.g. *abduction* → *abuse*).

2) Determining entailment direction

The integrated method provides a means to determine the entailment direction for distributional similarity candidates, which, by construction, are non-directional. Thus, amongst the (non-synonymous) distributional similarity candidate pairs classified as entailing, the direction of 73% was correctly identified.

3) Filtering co-hyponyms

The distributional similarity method is the more exhaustive, but less accurate of the two methods. By querying the web with both words of the candidate pair instantiated in the patterns, we obtain more focused and thus – more accurate – information about the pair itself than in the candidate acquisition stage. As an example, the integrated method successfully filters 65% of the non-entailing co-hyponym candidates, most of them originated in the distributional candidates, which constitutes a large portion (23%) of all correctly discarded pairs. Consequently, the precision of distributional similarity candidates approved by the integrated system was nearly doubled,

	Pattern-based	Distributional
False Positives	gap → hazard	chairman → advisor
	abuse → trauma	vice president → chairman
	management → field	shortfall → surplus
	penalty ↔ sentence	abuse → corruption
	management → issue	government → parliament
False Negatives (misses)	sentence → text	privatization ↔ privatisation
	belt → goods	trade deficit → gap
	chairman → role	federation → establishment

Table 8: Typical errors generated by the integrated method. Section 4.3.1 details our analysis of these errors.

demonstrating the additional information that patterns provide about distributionally similar pairs.

4) Filtering patterns-generated errors

The distributional method contributed to removing many of the wrong candidates suggested by the pattern-based method. For instance, the pair *goods - time* scored highly among the pattern-related features, due to many occurrences in sentences with more complicated syntactic dependencies than can be handled without deep parsing. This is the case, for example, in: “*The FTC’s fund balances with Treasury are carried forward until such time as goods or services are received and payment is made*”. Yet, this pair scored very low by GD04 which may have contributed to its correct removal. The idea is that in distributional similarity each relationship is more established, even if its type is not clearly determined. Thus, erroneous pattern-based relationships like the above are given low distributional similarity scores, helping in the elimination of these kinds of errors.

Another interesting observation from the development set was that for distributional pairs which were entailing in a single direction, we could quite precisely (80%) identify the direction of entailment simply by using the direction feature (feature score > 1). Usually, the direction that appeared more in patterns was the correct direction. While this alone seems as a simple method to identify the direction of pairs generated from the non-directional distributional methods, it first requires distinguishing the pairs that are entailing in a single direction from the complete set of candidates, including those entailing in both directions.

4.3.1 Error Analysis

Several error cases that were detected and categorized are detailed below. Two types of errors are listed in Table 8 – *false positive* errors – where the system wrongfully identified a pair as entailing, and *false negative* errors – where correct entailment pairs were classified as non-entailing, i.e., were missed.

1) Extraction-related Errors

We used several external applications for the purpose of syntactic processing. Errors created during these steps have propagated to the candidate extraction stage. Other syntactic errors occur due to the fact that snippets aren't always complete sentences. Sentences which are cut in the middle are not always correctly analyzed. For the extraction, following the idea that exploiting the redundancy of the web should compensate for the simplicity of the extraction, we preferred using simpler patterns rather than more accurate but more complex extraction rules. This, naturally, added some errors to our results. Possibly we could eliminate some errors by not using an NP-chunker which caused many extraction errors (mostly misses) by complicating the patterns required for extracting the targeted terms. Note that a POS Tagger might have been enough if we'd settle for single-word nouns, but as we did wish to extract multi-word terms, NP chunking was the processing level of choice. Lastly, our choice to allow flexible pattern structure by using the asterisks feature in web queries increased the variability of terms found, but inserted some noise into the data.

2) Ambiguous Dependencies

Another extraction-related error is due to more complex syntactic dependencies, which are not resolvable with an NP Chunker. When matching pairs in a pattern, we extracted the closest two noun-phrases within the pattern. This was correct in most cases, but not always, especially when a noun phrase of the pattern was constructed from two noun phrases separated by a preposition. In such cases, the head is usually in the first noun phrase and the pattern may refer to it. For example, in “*governors of central banks and other institutes*”, we would correctly extract *central bank* as entailing *institute*, but in “*governors of central banks and other high-level officials*”, we'd wrongfully identify *central-bank* as entailing *official*. This kind of error is most likely not be solved even if dependency parsing is applied, as parsers such as Minipar

(Lin, 1993) assign the same dependency structure to *central banks* and *institutes* in the first sentence and to *central banks* and *officials* in the second.

3) Context-dependent entailment

Many non-entailing candidate pairs were identified because they exhibit an entailment relation in a specific context. However they do not entail each other in their general meaning, and therefore fail to satisfy the lexical entailment's *word meaning entailment* criterion. Consider the following sentence: "*On this slide you will encounter such hazards as - Lava rivers, narrow paths, gaps in the ground, jumping lava and huge, massive rolling balls*". In this particular context, *gap* does constitute a *hazard*, but there is no single sense in which these words entail each other in their general meanings. A physical gap, like in the above context may also be between any two objects without creating a hazardous situation.

This type of error frequently happens for co-hyponyms and for distant hypernyms. For instance, the co-hyponyms *chairman* and *advisor* were wrongfully marked as entailing following sentences like: "*the company's chairman, a specialist financial advisor with 30 years of experience*". Additionally, distant hyponymy relationships often include generic terms like *issue*, *element* or *factor* as the hypernyms. In more than a few cases, candidates including such terms were erroneously marked as entailment pairs by our system. These terms are not easy to disregard automatically: Using a stop-words list to filter out these terms will not do as in some senses almost every such term may be specific. While *element* is generic in "*...must be interpreted more broadly to include such elements as inappropriateness, injustice and lack of predictability*", it is specific when referring to a chemical element or to one of the five elements in Chinese tradition. We checked whether marking "popular hypernyms" - terms that serve as hypernyms of many distinct hyponyms - as suspicious generic terms may help, but it turns out that these generic terms were not always more popular than other, valid ones.

This kind of errors is more typical to the pattern-based approach, which is sometimes permissive with respect to the relationship captured and may also extract candidates from a relatively small number of pattern occurrences.

4) Synonyms are infrequent in patterns

Synonyms tend to appear less frequently in patterns. Consequently, some synonymous pairs discovered by distributional similarity were rejected due to insufficient pattern matches. Anecdotally, some typos and spelling alternatives, like *privatization* ↔ *privatisation*, are also included in this category as they rarely co-occur in patterns.

5) Pattern Ambiguity

A large portion of errors is caused by pattern ambiguity. For example, the pattern "*NP₁, a|an NP₂*"⁸, ranked among the top IS-A patterns by (Pantel et al., 2004), can represent both apposition (entailing) as in, "*...The next evolution on top of the iPod is the **digital-media player, a device that plays multiple forms of content***" and a part of list of co-hyponyms (non-entailing), such as in "*the browser which comes with 120 search engines, all the major **media players, a DVD player, and an HTML and WAP editor built in ...***".

⁸ We note that in practice, the determiner ("a" or "an") is included by the chunker we used in the second NP.

5 Conclusion and Future Work

The main contribution of this thesis is a novel integration of the so far mostly isolated two primary approaches for lexical semantic acquisition. Our investigation highlights the complementary nature of the two approaches and the information they provide from the wealth of complementary information that is contained in corpora. Notably, it is possible to extract pattern-based information that complements the weaker evidence of distributional similarity, while the task of semantic relationships discovery is best performed when both methods are utilized. Supervised learning was found very effective for integrating the different information types, yielding noticeably improved performance. Indeed, our analysis reveals that the integrated method helps eliminating many error cases typical to each method alone. The integrated approach was applied here to lexical entailment. We suggest that this line of research may be investigated further to enrich and optimize the learning processes and to address additional lexical semantic relationships.

Our study highlighted the potential in combining the two methods and constructing meaningful features for them. Yet, the results we achieved may be improved in many respects. Obviously, downloading more snippets from the web seems like a promising way to improve figures. By downloading more snippets, more evidence is obtained, statistics become more significant and the number of candidates is likely to increase. Relative recall, as we calculated in this research, may not necessarily rise, but the overall yield will, allowing us to further prefer precision on recall's expense, making the results even more valuable. Using a larger corpus for calculating distributional similarity scores and judging more examples to enlarge the training set are also likely to improve the classifier performance. Apart from technical issues, such as refinement of syntax processing tools and extraction patterns, several research directions are suggested:

- 1) Geffet and Dagan (2005) used the web to obtain information about pairs acquired by distributional similarity, achieving a higher precision than ours for these pairs. It may be beneficial to use their method to improve the acquisition of distributional similarity pairs.
- 2) **Queries bootstrapping:** Not all the queries we used were equally valuable, as some showed better productivity of entailing pairs or better accuracy than others. While we want to keep the flexibility of the queries in order to find multi-word noun

phrases and to find terms within such phrases, we can identify the more effective queries and use them more extensively, as well as filtering out the more noisy queries. We can then repeat the feature construction stage while downloading more snippets per each high-quality query, thus improving our yield and precision.

- 3) **Evaluation:** While Kappa was reasonable, we believe that evaluation could have been improved had the evaluators been given example sentences for the pairs they evaluated. The reason we avoided giving examples in the first place was to avoid bias between pairs that had example sentences found for them in our queries and pairs that didn't. This last kind of pairs were in many cases "reversed pairs", i.e. pairs that were created (rather than found) by reversing the direction of pairs discovered by the pattern-based method. The reversing was necessary since for distributional pairs both directions were considered as the method is non-directional. Lack of examples correlated with incorrect pairs, therefore, we did not want to create a judgmental bias by providing some pairs with examples and some without them.

We believe that by finding an effective and objective method for providing example sentences for all evaluated pairs, we could improve inter-judge agreement and thus improve the overall performance of the system, as a more accurate training model will be created.

- 4) **Global tree structure:** In the pattern-based features we used, we looked at the direct relationships between the two terms. It may be possible to obtain more information by trying to place the terms in their locations in a global, WordNet-like tree of relationships. For example, if we discover that two terms have a common hypernym, it may mean that the terms are semantically related. This piece of information may hint that the words are similar in their meanings, but for lexical entailment purposes, this is no clear-cut, as two terms sharing a hypernym may be synonyms or hyponyms (entailing), or co-hyponyms (not entailing). However, what is expected to be a more significant feature is the existence of common hyponyms. In this case, the relation may only be entailing – as the terms can only be synonyms or in hyponymy relation.

References

- Ageev, Mikhail S. and Boris V. Dobrov. 2003. Support Vector Machine Parameter Optimization for Text Categorization Problems. In Proceedings of ISTA-03. Kharkiv, Ukraine.
- Berland, Matthew and Charniak, Eugene. 1999. Finding Parts in Very Large Corpora. In Proceedings of ACL-99. Maryland, USA.
- Brin, Sergey. 1998. Extracting Patterns and Relations from the World Wide Web. In WebDB Workshop at 6th International Conference on Extending Database Technology, EDBT-98. Valencia, Spain.
- Caraballo, Sharon A. 1999. Automatic Construction of a Hypernym-Labeled Noun Hierarchy from Text. In Proceedings of ACL-99. Maryland, USA.
- Chklovski, Timothy and Patrick Pantel. 2004. VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations. In Proceedings of EMNLP-04. Barcelona, Spain.
- Curran, James R. 2005. Supersense Tagging of Unknown Nouns using Semantic Similarity. In Proceedings of ACL-05. Ann Arbor, Michigan, USA.
- Dagan, Ido, Oren Glickman and Bernardo Magnini. 2005. The PASCAL Recognizing Textual Entailment Challenge. In Proceedings of the PASCAL Challenges Workshop for Recognizing Textual Entailment. Southampton, U.K.
- Davidov, Dmitry and Ari Rappoport. 2006. Efficient Unsupervised Discovery of Word Categories Using Symmetric Patterns and High Frequency Words. In Proceedings of COLING/ACL-06. Sydney, Australia.
- Etzioni, Oren, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D.S. Weld, and A. Yates. 2004. Web-Scale Information Extraction in KnowItAll. In Proceedings of WWW-04. NY, USA.
- Fellbaum, Christiane, editor. 1998. WordNet: An Electronic Lexical Database. MIT Press, Cambridge, MA USA.
- Finkelstein-Landau, Michal and Emmanuel Morin. 1999. Extracting Semantic Relationships between Terms: Supervised vs. Unsupervised Methods. In Proceedings of International workshop of Ontological Engineering on the Global Information Infrastructure. Dagstuhl Castle, Germany.

- Geffet, Maayan and Ido Dagan. 2004. Feature Vector Quality and Distributional Similarity. In Proceedings of COLING-04. Geneva, Switzerland.
- Geffet, Maayan and Ido Dagan. 2005. The Distributional Inclusion Hypothesis and Lexical Entailment. In Proceedings of ACL-05. Michigan, USA.
- Geffet, Maayan. 2006. PhD Dissertation. School of Computer Science and Engineering at the Hebrew University of Jerusalem.
- Girju, Roxana, Adriana Badulescu and Dan Moldovan. 2003. Learning Semantic Constraints for the Automatic Discovery of Part-Whole Relations. In Proceedings of HLT-NAACL-03. Edmonton, Canada.
- Harabagiu, Sandra and Andrew Hickl. 2006. Methods for Using Textual Entailment in Open-Domain Question Answering. In Proceedings of COLING/ACL-06. Sydney, 2006
- Harris, Zelig S. 1968. Mathematical Structures of Language. Wiley.
- Hearst, Marti. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In Proceedings of COLING-92. Nantes, France.
- Hovy, Eduard, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2002. Using Knowledge to Facilitate Factoid Answer Pinpointing. In Proceedings of COLING-02. Taipei, Taiwan.
- Joachims, Thorsten. 1999. Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press.
- Kwong, Oi Yee. 2001. Word Sense Disambiguation with an Integrated Lexical Resource. In Proceedings of the NAACL-01 Workshop on WordNet and Other Lexical Resources. Pittsburgh, Pennsylvania, USA.
- Lin, Dekang. 1993. Principle-Based Parsing without Overgeneration. In Proceedings of ACL-93. Columbus, Ohio, USA.
- Lin, Dekang. 1998. Automatic Retrieval and Clustering of Similar Words. In Proceedings of COLING/ACL98. Montreal, Canada.
- Lin, Dekang, Shaojun Zhao, Lijuan Qin, and Ming Zhou. 2003. Identifying synonyms among distributionally similar words. In Proceedings of IJCAI-03. Acapulco, Mexico.

- Lin, Jing. 2006. Using Distributional Similarity to Identify Individual Verb Choice. In Proceedings of the Fourth International Natural Language Generation Conference. Sydney, Australia.
- Lindén, Krister and Jussi Piitulainen. 2004. Discovering Synonyms and Other Related Words. In Proceedings of CompuTerm-04. Geneva, Switzerland.
- Mirkin, Shachar, Ido Dagan and Maayan Geffet. 2006. Integrating Pattern-Based and Distributional Similarity Methods for Lexical Entailment Acquisition. In Proceedings of COLING/ACL-06 Main Conference Poster Sessions. Sydney, Australia.
- Moldovan, Dan and Rada Mihalcea. 2000. Using WordNet and Lexical Operators to improve Internet Searches. IEEE Internet Computing, vol. 4 no. 1, January 2000.
- Moldovan, Dan, Adriana Badulescu, Marta Tatu, Daniel Antohe and Roxana Girju. 2004. Models for the Semantic Classification of Noun Phrases. In Proceedings of the HLT/NAACL-04 Workshop on Computational Lexical Semantics. Boston, MA.
- Negri, Matteo. 2004. Sense-based Blind Relevance Feedback for Question Answering. In SIGIR-2004 Workshop on Information Retrieval For Question Answering (IR4QA). Sheffield, UK.
- Pantel, Patrick and Dekang Lin. 2002. Discovering Word Senses from Text. In Proceedings of SIGKDD-02. Edmonton, Canada.
- Pantel, Patrick, Deepak Ravichandran, and Eduard Hovy. 2004. Towards Terascale Semantic Acquisition. In Proceedings of COLING-04. Geneva, Switzerland.
- Pantel, Patrick and Deepak Ravichandran. 2004. Automatically Labeling Semantic Classes. In Proceedings of HLT/NAACL-04. Boston, MA.
- Pasca, Marius and Sanda Harabagiu. 2001. The Informative Role of WordNet in Open Domain Question Answering. In Proceedings of the Workshop on WordNet and Other Lexical Resources. Pittsburgh, PA USA.
- Ravichandran, Deepak and Eduard Hovy. 2002. Learning Surface Text Patterns for a Question Answering System. In Proceedings of ACL-02. Philadelphia, PA.
- Romano, Milen Kouylekov, Idan Szpektor, Ido Dagan and Alberto Lavelli. 2006. Investigating a Generic Paraphrase-based Approach for Relation Extraction. In Proceedings of EACL-06. Trento, Italy.
- Rosso, Paolo, Edgardo Ferretti, Daniel Jimenez and Vicente Vidal. 2004. Text Categorization and Information Retrieval Using WordNet Senses. In Proceedings of the Second Global WordNet Conference. Brno, Czech Republic.

- Schölkopf, Bernhard. Statistical learning and kernel methods. 2000. MSR-TR 2000-23, Microsoft Research.
- Shinzato, Kenji and Kentaro Torisawa. 2004. Acquiring Hyponymy Relations from Web Documents. In Proceedings of HLT/NAACL-04. Boston, MA.
- Szpektor, Idan, Hristo Tanev, Ido Dagan and Bonaventura Coppola. 2004. Scaling Web-based Acquisition of Entailment Relations. In Proceedings of EMNLP-04. Barcelona, Spain.
- Sundblad, Håkan. Automatic Acquisition of Hyponyms and Meronyms from Question Corpora. 2002. In Proceedings of the ECAI-02 Workshop on Natural Language Processing and Machine Learning for Ontology Engineering. Lyon, France.
- Turney, Peter D. 2006. Expressing Implicit Semantic Relations without Supervision. In Proceedings of COLING/ACL-06. Sydney, 2006
- Voorhees, Ellen M. 1994. Query Expansion Using Lexical-Semantic Relations. In Proceedings of SIGIR-17. Dublin, Ireland
- Yangarber, Roman, Ralph Grishman, Pasi Tapanainen and Silja Huttunen. 2000. Automatic Acquisition of Domain Knowledge for Information Extraction. In Proceedings of COLING-00. Saarbrücken, Germany.
- Zhao, Shaojun. 2004. Named Entity Recognition in Biomedical Texts using an HMM Model. International Joint workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP-04). Geneva, Switzerland.

Appendix A: Web Pattern-based Extraction Module

This appendix provides more details about the web pattern-based extraction package presented in Section 3 and is included for technical competence.

Overall, the module includes three main steps performed for every term or pair of terms, as listed hereunder. Details of each of the steps are given immediately below.

1. Query creation
2. Snippet processing - downloading, cleaning, filtering and syntactic processing
3. Relationship extraction

1. Query Creation

Web search is performed in our method at two stages. First, when given a target term for which we search for candidate entailing/entailed terms; and second - when we wish to collect information from the web for a candidate pair. Hence, the queries construction method must support queries with a single term and a variable, and with two terms. Terms may contain several words and we wish to find instances where the noun is not necessarily in its unmodified form in the pattern. In addition, queries are a valuable resource, since – for legal reasons – the number of queries per day one can submit to a commercial search engine is limited. All these considerations were taken into account in designing the queries construction module of this package.

Queries were submitted to Google’s search engine through Google API. Therefore, the queries are constructed using the engine’s syntax, while complying with some restrictions posed by the API package, such as a limit to the maximal number of words per query. We exploited a feature of the search engine which allows using an asterisk instead of any single word (some stop words are excluded from this count, such as *a*, *the* etc.). Up to three consecutive asterisks are supported.

At the candidate acquisition stage, given a single term, the second term – a variable – is represented in the query by an asterisk. Suppose, for example, that the target word is *loan* and we wish to find entailment pairs containing it. We create queries for each of the patterns with the word *loan*, in either of the two roles – as entailing and as entailed, where the second slot in the patterns is not instantiated. Some of the created queries are:

“ especially *** (loan OR loans)”*

" especially (loan OR loans)"*
" especially * (loan OR loans)"*
" especially ** (loan OR loans)"*
" including * (loan OR loans)"*
" like * (loan OR loans)"*
*"such ** as (loan OR loans)"*
*"such * as *** (loan OR loans)"*

In the above example queries, *loan* is in the entailing position. The following examples show it in the entailed position:

*"(loan OR loans) including *"*
*"(loan OR loans) such as *"*
*** or other ** (loan OR loans)"*
*** or other *** (loan OR loans)"*

As shown above, every possible and relevant combination of zero to three asterisks is used to allow the word to appear as sub-part of noun phrases and not necessarily as an unmodified noun. This extension over previous works increases the variability of terms that can be extracted by the system, and has the potential to increase the yield as well. However, since in many cases the words before the target word do not consist of modifiers, noise increases as well and errors may be inserted, especially since the syntactic processing isn't perfect. Note that the place holder for the variable is always present.

It should be noted that using asterisks within Google's queries is merely a shortcut to increase the productivity of our queries and minimize the number of submitted queries. Has Google changed this mechanism we could simply search for instances appearing in patterns based only on the lexical segments of the pattern, e.g. *"such as" "(loan OR loans)"*, which will still produce relevant results, but will require many more queries to achieve the same yield. Other than that, the queries contain no other engine-specific features. On average, this method extracted more than 3300 relationship instances for every 1MB of downloaded text, almost a third of them contained multi-word terms.

In the second mode, when we wish to collect information about a pair of term, both slots of the pattern are instantiated. For example, for the pair *central bank* - *agency*, here are some of the created queries:

*"(agency OR agencies) such as * central (bank OR banks)"*

*"(agency OR agencies) especially ** central (bank OR banks)"*

*"(agency OR agencies) including ** central (bank OR banks)"*

*"agencies are ** central banks"*

*"central bank is (a OR an) *** agency"*

*"central (bank OR banks) or other ** (agency OR agencies)"*

Singular vs. Plural

The meaning of a pattern may depend on whether the noun appearing in it is in singular or in plural form. Generally speaking, when looking for entailment relations, it is more common to find both entailing and entailed nouns in plural forms, as in the case of: *“Non-education loans, such as home equity loans and lines of credit, may also be worth considering”*. This is more obvious in some patterns than others, as in NP_1 and other NP_2 we will hardly find the second noun phrase in a singular form. However, it is impossible to use the plural form alone for two reasons: First, in some patterns both options are frequent. Second, for some nouns, and especially mass nouns, the singular form is commonly used to express plural or the plural form is hardly in use. *Fruit*, for example, is very commonly used to express plural, along with *Fruits*; *electricity* is by far more frequently used than *electricities*. Since we cannot set a clear rule for the behavior and cannot expect to receive such information for each target noun, we use both singular and plural form of the noun in the queries, except in cases when surely only one option is possible (e.g. NP_1 are NP_2). This holds the risk of reducing the accuracy of the system, as more sentences with wrong entailment relations are retrieved, but through the further steps we apply, we expect this problem to be mostly eliminated and the benefit to be higher than the risk.

To practically produce the singular/plural term given only one of the forms, as would happen when extracting the term from the web, we implemented a converter that uses linguistic conversion rules, WordNet list of irregular nouns and heuristics based on web statistics when the previous two options do not suffice.

2. Snippet Processing

For each query submitted to the search engine, we download a predefined configurable number of snippets. Snippets are used for a practical reason – we do not need to download and process the entire document for a single instance of the pattern we’re after. The drawback in using snippets is that many times they contain partial sentences, decreasing the accuracy of the syntactic processing.

Each downloaded snippet is cleaned from HTML tags and is converted to regular text format. Then, using a word splitter and a sentence segmenter from The University of Illinois at Urbana-Champaign, we tokenize the snippets and split them into sentences. Each sentence that does not contain the target term/s is deleted. Then, all duplicate sentences are deleted as well.

Syntactic Processing

Using OpenNLP Part of Speech Tagger and NP-Chunker, we processed each of the sentences. Here’s an example of a sentence retrieved for *war*, processed up to shallow parsing:

" [PP In/IN] [NP Africa/NNP] ,/, [NP trans-border/NN conflicts/NNS] [VP including/VBG] [NP wars/NNS] [VP have/VBP] [NP an/DT important/JJ impact/NN] [PP on/IN] [NP the/DT use/NN] [PP of/IN] [NP regional/JJ infrastructure/NN] ;/:"

3. Relationship Extraction

Pattern-specific regular expressions are used to extract relationships from the chunked sentences. For example, from the above sentence which was retrieved for the word *war* with a query produced for the pattern NP_1 including NP_2 , the relation $war \rightarrow conflict$ is extracted using regular expression defined for the pattern. This pattern is represented by the following regular expression grammar (shown here in partial, see full grammar in Appendix C):

- i. NP_1 including $NP_2 = "NPWithPossCommaExp$ includingExp $NPListExp"$
- ii. $NPWithPossCommaExp = ((NPWithCommaExp)|(NPExp))$
- iii. $NPListExp = ((NPListEx1)|(NPListEx2))$
- iv. $NPListEx2 = (NPExp ,/,) * NPandNPExp$

v. *includingExp* = $\langle (PP|VP) \textit{including}/VBG \rangle$ ⁹

...

An extracted relation must comply with the following constraints:

- i. Neither term is tagged as a proper noun
- ii. One term cannot be a modified version of the second, as we consider these to be trivial entailment pairs. For example, pairs like *commercial real estate loan – loan* are discarded.

iii. Neither is included in the following list of stop words:

a, an, the, your, yours, his, her, hers, our, ours, their, theirs, this, that, them, these, those, it, its, it's, which, whose, some, other, thing, something, one, such

⁹ The original markers of a chunk's beginning and end are “[“ and “]”. Before applying the extraction we replaced them with “<” and “>” respectively for easier construction of regular expressions since “[“ and “]” are reserved symbols in regular expression syntax.

Appendix B: POS Tagger and NP Chunker Reference

OpenNLP's POS Tagger uses the Penn Treebank POS tag set, which is listed in Table 9 below. The NP Chunker in its turn uses notations such as *NP* (Noun Phrase), *VP* (Verb Phrase) and *PP* (Prepositional Phrase) to mark larger segments (chunks) of the text, e.g. *[NP no/DT outstanding/JJ book/NN debts/NNS]* indicating that the whole segment is a noun phrase. The complete notation and documentation is found on the Penn Treebank Project's website at <http://www.cis.upenn.edu/~treebank/>

Tag	Description	Example
CC	Coordinating conjunction	and, &
CD	Cardinal number	502, million
DT	Determiner	all
EX	Existential there	there
FW	Foreign word	etc
IN	Preposition or subordinating conjunction	for
JJ	Adjective	foreign
JJR	Adjective, comparative	more
JJS	Adjective, superlative	most
LS	List item marker	1.
MD	Modal	cannot
NN	Noun, singular or mass	bank
NNS	Noun, plural	accounts
NNP	Proper noun, singular	Finland
NNPS	Proper noun, plural	Fuels
PDT	Predeterminer	all/PDT those/DT factors/NNS
POS	Possessive ending	'
PRP	Personal pronoun	you
PRP\$	Possessive pronoun	its, their
RB	Adverb	approximately
RBR	Adverb, comparative	worse, less, more
RBS	Adverb, superlative	most
RP	Particle	out
SYM	Symbol	=

TO	to	to
UH	Interjection	oh, please
VB	Verb, base form	buy
VBD	Verb, past tense	chose
VBG	Verb, gerund or present participle	setting
VBN	Verb, past participle	bought, known
VBP	Verb, non-3rd person singular present	are
VBZ	Verb, 3rd person singular present	is
WDT	Wh-determiner	that, what, whatever
WP	Wh-pronoun	which, who, whom
WP\$	Possessive wh-pronoun	whose
WRB	Wh-adverb	when

Table 9: Penn Treebank POS Tagger reference.

Appendix C: Patterns Grammar

Table 10 lists the regular expressions used by our method in order to construct the extraction patterns grammar. Table 11 shows how the patterns' expressions were constructed from these building blocks. The extraction stage, applied on chunked sentences, was designed to trade off precision and recall in the extraction by handling some of the common chunker errors, while not attempting to cover all cases. As mentioned, while compiling it we had in mind the idea that when using the web it might not be necessary to extract information from complex text structures, but rather settle for simpler text while relying on the scale and redundancy of the web (Etzioni, 2004).

	Name	Regular Expression
1	<i>NPExp</i>	<NP [^>]* >
2	<i>suchAsExp</i>	<PP such/JJ as/IN >(:/)?
3	<i>suchExp</i>	<NP (s S)uch/JJ [^>]*>
4	<i>asExp</i>	<PP as/IN >
5	<i>NPListEx1</i>	<i>NPExp</i> ((/, <i>NPExp</i>){0,10} (and or)/CC <i>NPExp</i>)?
6	<i>NPandNPExp</i>	<NP [^>]* (and or)/CC [^>]* >
7	<i>NPListEx2</i>	(<i>NPExp</i> /,)* <i>NPandNPExp</i>
8	<i>NPListExp</i>	((<i>NPListEx1</i>)(<i>NPListEx2</i>))
9	<i>NPWithCommaExp</i>	<NP [^>]* > /,
10	<i>NPWithPossCommaExp</i>	((<i>NPWithCommaExp</i>)(<i>NPExp</i>))
11	<i>includingExp</i>	<(PP VP) including/VBG >
12	<i>especiallyNPExp1</i>	(and/CC)?<ADVP especially/RB > <i>NPExp</i>
13	<i>especiallyNPExp2</i>	(and/CC)?<NP especially/RB [^>]* >
14	<i>especiallyNPExp</i>	((<i>especiallyNPExp1</i>)(<i>especiallyNPExp2</i>))
15	<i>likeExp</i>	<PP like/IN >
16	<i>advExp</i>	(([/^]* /RB)(<ADVP [^/]* /RB >))
17	<i>advKnownAsExp</i>	((<VP (<i>advExp</i>)?known/VBN >)((<i>advExp</i>)?<VP known/VBN >)) <PP as/IN >
18	<i>isExp</i>	<VP is/VBZ >
19	<i>areExp</i>	<VP are/VBP >
20	<i>aNPExp</i>	<NP (a an)/DT [^>]* >
21	<i>andExp</i>	and/CC
22	<i>orExp</i>	or/CC
23	<i>otherNPExp</i>	<NP other/JJ [^>]* >

Table 10: Extraction patterns regular expressions building blocks. The symbols “<” and “>” are the markers for chunk beginning or end respectively.

	Pattern name	Pattern
1	$NP_1 \downarrow$ <i>such as</i> $NP_2 \uparrow$	$NPWithPossCommaExp \downarrow$ <i>suchAsExp</i> $NPListExp \uparrow$
2	<i>Such</i> $NP_1 \downarrow$ <i>as</i> $NP_2 \uparrow$	<i>suchExp \downarrow asExp</i> $NPListExp \uparrow$
3	$NP_1 \uparrow$ <i>or other</i> $NP_2 \downarrow$	$NPListExp \uparrow$ <i>orExp</i> <i>otherNPExp \downarrow</i>
4	$NP_1 \uparrow$ <i>and other</i> $NP_2 \downarrow$	$NPListExp \uparrow$ <i>andExp</i> <i>otherNPExp \downarrow</i>
5	$NP_1 \downarrow$ <i>ADV known as</i> $NP_2 \uparrow$	$NPWithPossCommaExp \downarrow$ <i>advKnownAsExp</i> $NPExp \uparrow$
6	$NP_1 \downarrow$ <i>especially</i> $NP_2 \uparrow$	$NPWithPossCommaExp \downarrow$ <i>especiallyNPExp \uparrow</i>
7	$NP_1 \downarrow$ <i>like</i> $NP_2 \uparrow$	$NPWithPossCommaExp \downarrow$ <i>likeExp</i> $NPListExp \uparrow$
8	$NP_1 \downarrow$ <i>including</i> $NP_2 \uparrow$	$NPWithPossCommaExp \downarrow$ <i>includingExp</i> $NPListExp \uparrow$
9	NP_1 -sg \uparrow <i>is</i> (<i>a</i> OR <i>an</i>) NP_2 -sg \downarrow	$NPExp \uparrow$ <i>isExp</i> <i>aNPExp \downarrow</i>
10	NP_1 -sg \uparrow (<i>a</i> OR <i>an</i>) NP_2 -sg \downarrow	$NPWithCommaExp \uparrow$ <i>aNPExp \downarrow</i>
11	NP_1 -pl \uparrow <i>are</i> NP_2 -pl \downarrow	$NPExp \uparrow$ <i>areExp</i> <i>NPExp \downarrow</i>

Table 11: Extraction patterns grammar. \downarrow indicates that the expression is assigned as entailed term, \uparrow indicates it is assigned as entailing. Note that for pattern 5, 9-11 it is not obvious which is the direction of entailment, and the direction marked here reflects the more common use of the pattern. In any case, the

Appendix D: SVM Training Procedure

Following (Hsu et al.¹⁰; Ageev and Dobrov, 2003) and our own experiments, we performed a procedure aimed at creating an optimized classification model. The same procedure was applied to each of the training sets we created a model for. The steps used in the procedure are detailed below.

Scaling

The features scores of the training set are scaled to a 0 to 1 range, separately for each of the 16 features. The purpose is to simplify numerical calculations and to avoid having a feature which ranges in higher values becoming more dominant than features of lower value ranges. The factors by which each feature score of the training set was multiplied were used to multiply the respective feature scores of the test set.

Kernel Function

We used the Radial Basis Function (RBF) kernel for the SVM classifier. RBF has produced better results in the optimization process than linear kernel and - in comparison to the polynomial kernel - has only one parameter which requires tuning - *gamma*, and therefore is easier to optimize. RBF is defined as follows:

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2), \gamma > 0$$

where x_i and x_j are (vector) instances of the input and γ defines the width of the RBF.

$\|x_i - x_j\|$ denotes the Euclidian distance between the two data points, defined as

$$\|x - y\| = \sqrt{\sum_{k=1}^n (x_k - y_k)^2}, n \text{ being the size of the vector, or - in our case - the number of}$$

features. See Figure 1 for an illustrated example of RBF kernel.¹¹

The idea of SVM is to map the input data into a higher dimension space (the feature space) where the data can be separated by a hyperplane. A kernel function can be viewed as a similarity measure between each two instances of the data. Using kernel functions, it is

¹⁰ Available at: <http://citeseer.ist.psu.edu/689242.html>

¹¹ Image source: SIGIR 2003 Tutorial on Support Vector and Kernel Methods by Thorsten Joachims available from: <http://www.cs.cornell.edu/courses/cs578/2005fa>

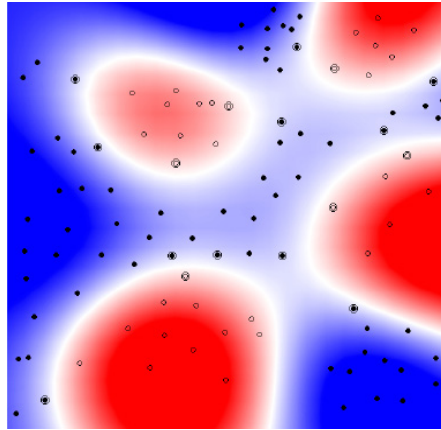


Figure 1: example of classification using the RBF kernel. Data points within the

possible to calculate the separating hyperplane without actually mapping the input into the higher space (Schölkopf, 2000).

Parameter Selection

Using *grid search* on 3 parameters, we chose the settings which produced maximal F1 result using cross validation on the training set.

Grid search is a naïve method to find optimal values of parameters: After setting minimal and maximal values, we increment the value of each parameter in its turn and check which combination produces the best results.

The parameters optimized are as follows:

1. RBF *gamma* parameter: Defines the RBF width.
2. SVM^{light} *c* parameter: Trade-off between training errors and margin. The smaller *c* is, the larger the margin is, and training errors increase. This parameter determines the weight of each of two error types: False positive errors, increased as the margin grows, and false negative errors (misses), which grow as the margin shrinks. Hence this parameter is used for trade off between precision (associated with the first error type) and recall (associated with the second).
3. SVM^{light} *j* parameter: This parameter is a cost-factor by which training errors on positive examples outweigh errors on negative examples, providing a way for compensating unbalanced training data. In our experiments *j*'s optimal value was found to be 2. Indeed, we had 1.92 more negative examples than positive ones.

To reduce the complexity of the search, the first two parameters were optimized first. Then the best value for *j* was found based on their optimized values.

10-Fold Cross Validation & Micro-Averaging F1

For the purpose of parameter optimization, we performed a 10-fold cross-validation on the training set and chose the parameters that yielded the best micro-averaged F1 score. In this 10-cycle process both training and test are performed on the training set: The complete training set is randomly divided into 10 subsets, each acts once as a test set and participates 9 times in a training set which consists of 9 subsets. The parameter setting achieving the best F1 score from the 10 cycles is chosen for the classification.

In our case, each of the 10 sets participating in the process consisted of all the pairs of a single input term. The reason we did not group together all candidate pairs of all terms and then randomly picked a tenth of them for each set is we cannot assume that the method, when given a target term will have prior knowledge about this term in the form of previously acquired pairs for it. Sampling from a set of all pairs, this is likely to happen, i.e. pairs of the same input term are likely to be included in both the training and test sets.

In micro averaging, the numbers of correct (true positives and true negatives) and incorrect (false positive and false negative) pairs are accumulated from the results of all cycles. Then, precision, recall and F1 are calculated from these sums. This measure was preferred over macro-averaging, where each cycle's F1 score is calculated separately and the scores are eventually averaged, due to the variability of the input terms. The fact that the acquisition task for some terms is easier than it is for others is reflected not only in the precision of the acquisition for the input term, but also in the number of pairs acquired for it. If we consider each cycle's test set separately, we give equal weight to sets of substantially different sizes. In a large training set, the sets size might have been more balanced, but in our setting - where the cross-validation test set included pairs of a single input term in each cycle – F1 scores of each cycle separately varied significantly, reflecting the acquisition difficulty of a single term. Micro-averaging gives equal weight to each acquired pair rather than to each input term, thus eliminating this problem.¹²

It should be noted that due to annotation resources limitation the training set was also used as development set. While it is best to use separate sets for these two purposes, it does not undermine the correctness of the evaluation scheme.

¹² Running cross-validation with less than 10 cycles left the test sets unbalanced, while adding complexity to the process as it required repeating it several times, with different sampling of the sets.