Modeling and Selection of Context for Better Synonym Acquisition

Masato Hagiwara

Abstract

Lexical knowledge is one of the most fundamental yet important resources for natural language processing, having a broad range of applications such as query expansion for information retrieval and automatic thesaurus construction. However, construction and maintenance of lexical knowledge by hand are costly tasks. Therefore, they should be technically supported or fully automated.

Among various kinds of lexical relations, synonym relation is frequently used as the basis in many applications. To automatically detect synonym relations from corpora, a concept called *distributional hypothesis*, which states that semantically similar words tend to share similar context, has been utilized. *Distributional similarity* is then computed from the extracted context.

However, although there have been a number of researches which focused on the similarity computation itself and the use of *context* is actually the essence of distributional similarity, the importance of context has been surprisingly underestimated so far. Firstly, very few studies have ever paid attention to the formalization or comparison of effective context for synonym acquisition, let alone the extension. However, how to construct and extend the context is the most fundamental issue for distributional similarity. Secondly, the effective context for synonym acquisition has only been intuitively determined, although the context used for synonym acquisition greatly influences the performance of the task. Thirdly, past studies have paid attention to the simple vector space model almost exclusively. However, more complex, semantic representation of context must be beneficial to synonym acquisition.

In response to these problems, we, in in this thesis, address the issues of formalization, extension, selection, and modeling of context in order to achieve better synonym acquisition from large corpora, which have been all unsolved or underestimated in the literature of synonym acquisition and distributional similarity. More specifically, the contribution of this thesis is three-fold — (1) As for the context representation and extension problem, we propose a new method to formalize and extend the conventional dependency-based context through the use of indirect dependency, and show its effectiveness. (2) As for the context selection problem, we propose three schemes to automatically select the extracted context types using statistical measures, and show that these schemes are effective in terms of their performance/cost trade-off. (3) As for the context modeling problem, we pursue the application of latent semantic models to formalize and model the context of words, or word-context co-occurrences, and show that these models can boost the synonym acquisition performance. Also, we propose supervised approaches to synonym acquisition, contextual feature-based classification and metric learning, and show that these both demonstrate the higher performance compared to conventional simpler models. Note that, although the methods in this thesis are based on English, they are not limited to any particular languages.

This thesis consists of a total of eight chapters. Chapter 1 is the introduction to this thesis, where we introduce the current issues and problems regarding distributional similarity and clarify the position of this thesis.

In Chapter 2, we introduce some basic concepts of distributional similarity, describe how to extract dependency-based context, and list a number of similarity measures and weighting functions, as the baseline for other experiments described in this thesis.

In Chapter 3, we actually apply the concepts introduced in Chapter 2 to the synonym acquisition task. To evaluate the performance, we introduce two evaluation measures, i.e., average precision (AP) and correlation coefficient (CC), which both use existing thesauri such as WordNet. As the preprocessing of experiments, we firstly investigate the effect of frequency cut-off to word and context types. We then compare similarity measures and weighting functions, and show that similarity measures which incorporate some type of normalization to the context vector or the probability distribution, namely cosine similarity, vector-based Jaccard coefficient, and Jensen-Shannon divergence, perform well. The experiment has also shown that PMI and t-test are among the best, meaning that the word-based normalization factor may have helped.

In Chapter 4, we introduce and apply two latent semantic models, i.e., LSI and PLSI, to synonym acquisition and compare their performance with the simple vector space model. Latent semantic models formalize the co-occurrences of words and contexts through latent semantics, and solve the problems of noise and sparseness. Although each model behaves quite differently, LSI and PLSI have achieved higher performance when coupled with their own suitable measures. We have also shown that the performance peaks when the number of the latent classes is around 100 to 150.

In Chapter 5, we formalize and extend the normal direct dependency to cover indirectly related words and enhance the contextual information for distributional similarity. The experiment shows that incorporating indirect dependency in addition to direct dependency is effective for the acquisition performance. We also compare the context representations for indirect dependency. The improvement is especially clear when fine-grained context representations are used.

In Chapter 6, we propose three schemes of context selection for distributional similarity: category-, type-, and co-occurrence based selection. In the experiment, these three selection schemes are compared, clarifying the characteristics of the schemes and the measures. It shows the effectiveness of the simplest category-based selection, while type- and co-occurrence based selection methods work well for both the word- and dependency-based context, showing that these method can be generally and flexibly used for any kinds of context and dimensionality/computational cost constraints.

In Chapter 7, two novel, supervised approaches to synonym acquisition are proposed. The first one is a classification model, where we propose context-based features called *distributional features*, which enabled to use both context-based features and *pattern-based features* to build fully integrated classifiers. The comparison experiment shows that the context-based features greatly increase the performance (more than 60% on F-1 measure) compared to distributional similarity-based methods. The other approach learns Mahalanobis distance, which is the generalization of Euclid distance. It significantly outperforms existing similarity metrics. Although we have

to resort to aggressive feature reduction to make it possible to apply the learning, the performance gain from the supervised learning is enough to offset the disadvantage and justify its usage in some applications.

In Chapter 8, we conclude this thesis, wrapping up the findings and results obtained from each chapter. We also present our possible future direction of this study.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

As the meaning of words is the fundamentals for understanding human languages, the knowledge of words is also essential for processing them by computers. Indeed, lexical knowledge has been one of the most fundamental yet important resources for natural language processing. A wide variety of lexical knowledge bases, including WordNet [Fellbaum, 1998], CYC [Lenat, 1995], and *Bunruigoihyo* [Kokken, 2004] have been constructed and utilized. However, the construction and maintenance of lexical knowledge by hand are difficult tasks. Therefore, they should be technically supported or fully automated.

Among various kinds of lexical relations, synonym relation is frequently used as the basis in a broad range of applications such as query expansion and indexing techniques for information retrieval [Crouch and Yang, 1992, Jing and Croft, 1994], text categorization [Budanitsky and Hirst, 2006], and automatic thesaurus construction [Kojima and Ito, 1995, Grefenstette, 1994]. It also plays an important role in constructing linguistic ontologies because the core techniques of ontology construction — relation assignment and taxonomy construction — would be almost impossible without synonym detection techniques.

To detect synonym relations from corpora, a concept called *distributional hypothesis* [Harris and (ed.), 1985], which states that semantically similar words share similar contexts, has been used so far. It suggests that words' relatedness can be estimated from the context they have, and the similarity computation can be automated using the context of words and some appropriate measures. The similarity obtained in this way is called *distributional similarity*. A number of studies [Hindle, 1990, Lin, 1998a, Curran and Moens, 2002a, Curran and Moens, 2002b, Geffet and Dagan, 2004] were conducted regarding how to compute and make the most of distributional similarity.

In these studies, the computation of distributional similarity usually follows these two steps: (1) analysis of corpora and extraction of useful context, and (2) computation of distributional similarity from the extracted context. As we can see, the use of *context* is actually the essence of distributional similarity. However, although there exist a number of research projects which focused on the similarity computation

itself, the importance of *context* has been often underestimated so far. Specifically, three important problems concerning context have been left unanswered: (a) what is context, (b) which context should be used, and (c) how context is used. We will discuss these issues in detail in the following.

As for the question (a), although some types of context, such as dependency-based context [Hindle, 1990, Lin, 1998a, Curran and Moens, 2002a] and word-based context [Ng and Lee, 1996, Lowe and McDonald, 2000, Curran and Moens, 2002b] are widely used, very few studies have ever paid attention to the formalization or comparison of effective context for synonym acquisition, let alone the extension. However, we believe that how to construct and extend the context is the most fundamental issue for distributional similarity. Therefore, the problems related to this question, *context representation and extension*, should be fully discussed.

One way to answer the question (b) is to formalize the question as the *context selection problem*. In fact, the effective context for synonym acquisition has only been intuitively determined so far [Curran and Moens, 2002a], and most of the studies simply ignored this issue and assumed that the context to be used was given a priori. However, we believe, and will actually show, that the context used for synonym acquisition greatly influences the performance of the task.

As for the question (c), one approach to this issue is to model the context or *co-occurrence* of words and context. Although past studies have paid attention to the simple vector space model almost exclusively, more complex, semantic representation of context must be beneficial to synonym acquisition. This question is reduced to the problem of *modeling of context*. Also, there can be another way to approach this issue — we can resort to a different formulation of a problem as long as the extracted context is used, which implies the use of advanced generative/discriminate models for synonym acquisition.

To sum up, the aim of this study is to address the formalization, extension, selection, and modeling problems of context in order to achieve better synonym acquisition from large corpora. More specifically, the contribution of this thesis is three-fold:

(a) As for the context representation and extension problem, we propose a new method to formalize and extend the conventional dependency-based context through the use of indirect dependency. We then show its effectiveness in the experiments.

(b) As for the context selection problem, we propose three schemes to automatically select the extracted context types using statistical measures. We then show that these schemes are effective in terms of their performance/cost trade-off.

(c) As for the context modeling problem, we pursue the application of latent semantic models to formalize and model the context of words, or word-context co-occurrences. We show that these models can boost the synonym acquisition performance. Also, we propose two novel approaches to synonym acquisition, supervised classification and metric learning. We then show that these both demonstrate the higher performance compared to conventional simpler models.

As the preliminaries for these topics, we also have to investigate some basic techniques, e.g., similarity measures, weighting functions, and frequency cut-off, as well as how to evaluate the result of the synonym acquisition task, all of which will be discussed in the early part of this thesis.

Because the issues we address in this thesis are mostly independent of each other, they can be combined and used in parallel, or integrated into other applications. Although this kind of combination is not the scope of this thesis, it can further boost the performance of synonym acquisition.

Note that we only target at English in this thesis, although none of the methods and models introduced and proposed in this thesis is limited or restricted to specific languages. They all can be applied to other languages, as long as the corpus in use is appropriately processed beforehand. The only language-dependent element of our methods could be the dependency-based context. However, a concept similar to dependency exists in most of the languages and the word-based context can always used instead, even if the former does not exist.

In the next section, we discuss some related work to each of the topics we are going to deal with, and clarify the position of this thesis. After that, we show the organization of this thesis at the end of this chapter.

1.2 Related Work

1.2.1 Distributional Similarity

Distributional similarity has proven to be a general method which can be applied to a broad range of applications, although its literature does not date back to decades ago. To the best of author's knowledge, Hindle [Hindle, 1990] pioneered this research area by extracting subject and object nouns with their verbs and automatically obtained related nouns. In addition to subject/object relationships, Lin [Lin, 1998a] extracted other dependency structures and showed the possibility that other relations were contributing. Beside these, there are a number of researches regarding distributional similarity [Curran and Moens, 2002a, Curran and Moens, 2002b, Geffet and Dagan, 2004]. However, one drawback of the distributional approaches is that these methods are all unsupervised approaches, thus appropriate weights and similarity measures have to be fixed beforehand. Appropriate measures depend largely on the actual task to which distributional similarity is applied. For this reason, measures and weights have been extensively compared so far [Lee, 1999, Curran and Moens, 2002a, Weeds et al., 2004], with varied results. We compare some of the similarity/distance measures as well as weighting functions in Chapter 2 as the preliminary of all the other experiments in this thesis.

Weeds and Wier's work [Weeds and Weir, 2003] is worth attention here — they proposed new measures, namely, precision and recall, considering the context matching process as information retrieval. They also proposed a combined approach, which is generalization of several distributional similarity measures with tunable meta-parameters. These meta-parameters were optimized on a development set. They

show that the proposed model outperformed specific conventional models. Although we can consider this as a step forward to a fully supervised distributional similarity-based approach, it does not use any machine learning-based techniques as we did in Chapter 7.

1.2.2 Modeling of Distributional Similarity

Several latent semantic models have been proposed so far, mainly for document indexing targeted at information retrieval. One of the most common is Latent Semantic Indexing (LSI) proposed by Deerwester et al. [Deerwester et al., 1990]. LSI is a geometric model based on the vector space model. It utilizes singular value decomposition of the document-term co-occurrence matrix, an operation similar to principal component analysis (PCA), to automatically extract major components that contribute to the indexing of documents. It can alleviate the noise and sparseness problems by a dimensionality reduction operation, that is, by removing components with low contributions to the indexing.

However, the LSI model lacks firm theoretical basis [Hofmann, 2001], thus it often requires some ad-hoc weighting such as inverse document frequency(idf) metric, although the optimality of idf, which is commonly used to weight elements, has yet to be shown [Mochihashi and Matsumoto, 2002].

On the contrary, PLSI, proposed by Hofmann [Hofmann, 1999], is a probabilistic version of LSI, where it is formalized that documents and terms co-occur through a latent variable. PLSI puts no assumptions on distributions of documents or terms, while LSI performs optimal model fitting, assuming that documents and terms are under Gaussian distribution [Hofmann, 2001]. Moreover, ad hoc weighting such as idf is not necessary for PLSI, although it is for LSI. PLSI is experimentally shown to outperform the former model [Hofmann, 1999]. The application of LSI and PLSI to synonym acquisition is only partially investigated, while we will fully discuss their contribution to the task as well as the related topics such as the effect of dimensionality reduction.

There are some other methods which incorporate latent semantics or reduce the original dimensionality. Latent Dirichlet Allocation (LDA) [Blei et al., 2003] is yet another generative model of documents, where the mixture parameter of latent semantics is drawn from the Dirichlet distribution. Bingham and Mannila proposed Random projection [Bingham and Mannila, 2001], which projects the document vectors onto a lower dimensional vector space using a stochastically created projection matrix. Semantic kernels [Kandola et al., 2002] is a propagation model on a barpitite graph, considering the semantic correlation between documents and words. It can alleviate the sparseness problem. Although these models can be effective for synonym acquisition, they are out of scope of this thesis because they are either computationally extensive or incapable of dealing with latent semantics.

1.2.3 Context Extension

So far, various kinds of contextual information have been used to acquire lexical relations based on the distributional hypothesis. One of the most fundamental yet well-performing ones is *word-based context* [Curran and Moens, 2002b], which considers words surrounding a target word as contexts. It has been extensively used in various applications including word sense disambiguation [Ng and Lee, 1996], word priming in cognitive science field [Lowe and McDonald, 2000], and synonym acquisition/thesaurus construction [Hagiwara et al., 2006]. The window size to capture contexts ranges from a few words on both sides to the entire paragraph or document, upon which the acquisition performance greatly depends. Although word-based context is simple and effective, it has some drawbacks, especially the data size problem. The method can easily include a large amount of non-relevant contexts and inflate the context size, making the similarity calculation computationally expensive [Hagiwara et al., 2006]. This is one of the biggest reasons why it is gradually taken over by richer kind of contexts.

On the other hand, more sophisticated and efficient context, *syntax-based context* [Pado and Lapata, 2007], has been employed in many lexical knowledge acquisition tasks. It uses words which have syntactic relation (mostly dependency structure) with the target word as contexts, sometimes along with their relation labels. For example, Hindle used subject and object relations to acquire synonyms automatically in his landmark paper in the field [Hindle, 1990]. This study accompanied a great deal of follow-ups in his wake — including [Pereira et al., 1993], where word clustering is conducted based on verbs and their objective nouns, and [Gamallo et al., 2001], where they investigated the difference of syntax-based contexts, and showed that the fine grained attributes improved the performance. Compared to word-based context, the effectiveness (or cost-effectiveness) of syntax-based context was shown by many studies [Curran and Moens, 2002b, Hagiwara et al., 2006, Pado and Lapata, 2007]. For example, Curran and Moens [Curran and Moens, 2002b] showed that syntax-based context could achieve the accuracy of more than 5 points higher with only two thirds of the original semantic space, compared to word-based context. These results show that syntax-based context is capable of choosing highly effective contexts. As such, some forms of extensions of syntax-based context are naturally worth consideration.

Syntax-based context is extended so that it includes not only words with direct relations but also ones with indirect relations, whose underlying ideas are essentially the same as the indirect dependency described in Chapter 5. Lin and Pantel [Lin and Pantel, 2001], for example, described the system which used *dependency path* to represent word relationships and extended the distributional hypothesis to extract equivalent paths for inference rule extraction of question answering task. However, their application is rather limited and its effect on distributional similarity and other kinds of tasks should be investigated. Pado and Lapata [Pado and Lapata, 2007] proposed a general framework to represent the variations of semantic space which incorporates indirect syntactic relations of words. Although they separated and formalized various parameter settings when constructing semantic spaces, they dealt with only one form of context representations (terminal word), i.e., how dependency

paths are mapped to basis in the semantic space, which we believe has critical impact on the performance. This will be fully discussed in Chapter 5.

1.2.4 Context Selection

As for the selection of contexts, a surprisingly small number of studies have paid attention to the extraction and selection of contexts, whereas a certain number of variations of context has been proposed and many studies were conducted as mentioned in the previous section. Earlier studies dealt with the selection problem, if any, only in a limited way. For example, Ruge [Ruge, 1997] proposed the use of dependency structure for automatic thesaurus construction and showed the result to be encouraging. However, neither the further investigation of dependency selection nor the comparison with other kind of contextual information is provided. Lin [Lin, 1998a] also used a wider range of grammatical relationship including modifications and showed the possibility that other kind of dependency relations in addition to subject and object was contributing, although it is still not clear what kind of relations affect the performance, or to what extent.

There are some studies which are first to pay attention to the comparison of various context categories for synonym acquisition [Curran and Moens, 2002b], [Hagiwara et al., 2006]. The former compared several context extractors such as window extractor and shallow- and deep-parsing extractor, while in the latter we compared various kinds of syntactical relations as finer-grained context categories. However, since these studies have conducted only a posteriori comparisons based on performance evaluation, we are afraid that these findings are somewhat limited to their own experimental settings which may not be applicable to completely new settings, e.g., one with a new set of contexts extracted from different sources. On the other hand, the type- and co-occurrence-based selection schemes proposed in Chapter 6 is applicable to any settings of context formalization.

As finer-grained context selection method, the concept of "frequency cutoff" used in a certain numbers of studies so far [Curran and Moens, 2002b], [Pado and Lapata, 2007], is worth mentioning. As we will discuss this in Chapter 3, this simply removes any word types (optionally, context types as well) whose frequencies are less than a fixed threshold. This is one of the simplest way to greatly reduce the computational cost while keeping the performance loss at minimum [Curran and Moens, 2002b], although it has been used with its effectiveness unquestioned, especially compared with other selection methods. Type-based selection scheme proposed in Chapter 6 is a generalization of this selection by frequency cutoff, where their effectiveness in terms of performance and computational cost is compared.

Curran and Moens [Curran and Moens, 2002a] focused on each co-occurrence of words and contexts, and suggested assigning an index vector of *canonical attributes*, i.e., a small number of representative elements extracted from the original vector, to each word. When the comparison is performed, canonical attributes of two target words are firstly consulted, then the original vectors are referred to only if the attributes have a match between them. This can be regarded as a context selection

method because important word-context co-occurrences are "selected" as an approximation of the original vector. However, it is not clear whether the condition for canonical attributes they adopted, i.e., that the attributes must be the most weighted subject, direct object, or indirect object, is optimal in terms of performance. Co-occurrence based selection presented in Chapter 6 is a generalization of this method. Also, the performance is evaluated in a more comprehensive way.

1.2.5 Pattern-based Lexical Knowledge Acquisition

Lexico-syntactic patterns in sentences have long been used, not only for synonym acquisition tasks but also for lexical knowledge acquisition tasks in general. Hearst [Hearst, 1993] pioneered this field, extracting hypernym/hyponym relations using syntactic patterns such as "such X as Y" and "Y and other X."

Even co-occurrence in a single sentence or in a window can be regarded as a generic type of "patterns," and Riloff and Shepherd [Riloff and Shepherd, 1997] used word co-occurrence within a window for extraction of semantic categories of words. Roark and Charniak [Roark and Charniak, 1998] used syntactic relations such as conjunctions and appositives to extract the same kinds of semantic categories.

Lexical patterns have also been intensely used for bootstrapping-based lexical acquisition methods. In order to extract semantic lexicons, i.e., unary relations of semantically related words, the *Basilisk* algorithm [Thelen and Riloff, 2002] relied on contextual evidence extracted from corpora, e.g. "*<subject> was arrested*", "*murdered <direct object>*", and "*collaborated with <pp object>*" for extracting terms belonging to the category *people*. The *Espresso* algorithm [Pantel and Pennacchiotti, 2006] also utilized patterns extracted from corpora, although it was designed for binary relation extraction. It also differed from *Basilisk* in that it exploited generic patterns, e.g., "X is a Y" for hypernym/hyponym relations and "X of Y" for meronym relations.

What is worth attention here is that supervised machine learning is easily incorporated with syntactic patterns. For example, Snow et al. [Snow et al., 2004] further extended Hearst's idea and built hypernym classifiers based on machine learning and syntactic pattern-based features, with success.

1.2.6 Supervised Approach

Supervised approach to lexical knowledge acquisition is gathering more and more attentions these days. Duplicate record detection has been an important problem in the database field. For example, Bilenko and Mooney [Bilenko and Mooney, 2003] proposed learnable string distance measures for this problem. One of the measures they proposed is based on the vector space model and SVM classification, using the features of word pairs built from common features of individual vectors. This corresponds to the "common features" described in Chapter 7.

Yu and Agichtein [Yu and Agichtein, 2003] worked on the detection of synonymous names of genes and proteins, utilizing a partially supervised approach, *Snowball*, and a fully supervised approach based on SVM classifier, using contextual patterns (i.e., connecting string, as done in *Espresso*).

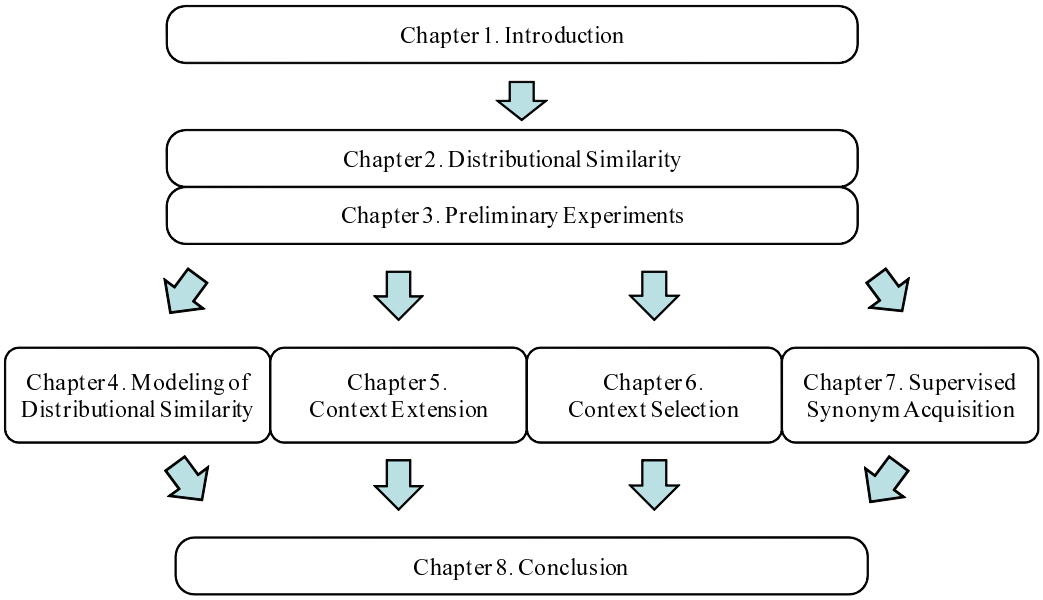

Figure 1.1: Organization of this thesis

Connor and Roth [Connor and Roth, 2007] is also worth mentioning here, who dealt with context sensitive lexical paraphrasing using a global, unsupervised, bootstrapped learning approach. Their focus is on how to deal with context sensitivity and how to utilize local context. Thus it does not directly compete with our method — the feature construction introduced here can be used together with their framework.

There are also some approaches to learn *metric* to capture the relatedness of words. Most previous metric learning methods learn the parameters of the Mahalanobis distance. Although the algorithms proposed in earlier work [Xing et al., 2003, Weinberger et al., 2006, Globerson and Roweis, 2006] were shown to yield excellent classification performance, these algorithms all have worse than cubic computational complexity in the dimensionality of the data. Because of the high dimensionality of our objects, we opted for information-theoretic metric learning proposed by [Davis et al., 2007]. This algorithm only uses an operation quadratic in the dimensionality of the data, which we will use in Chapter 7.

Other work on learning Mahalanobis metrics includes online metric learning [Shalev-shwartz et al., 2004], locally-adaptive discriminative methods [Hastie and Tibshirani, 1996], and learning from relative comparisons [Schultz and Joachims, 2003]. Non-Mahalanobis-based metric learning methods have also been proposed, though they seem to suffer from suboptimal performance, non-convexity, or computational complexity. Examples include neighborhood component analysis [Goldberger et al., 2004].

1.3 Organization of This Thesis

The organization of this thesis, which is illustrated in Figure 1.1, is as follows.

In Chapter 2, we introduce the basic concept of distributional similarity, showing examples and defining some basic terms. We then describe how to extract dependency-based context and to construct the simplest vector space model (VSM). Finally, we introduce some of similarity measures and weighting functions which have been proposed in the previous work, to be used as the baseline throughout this thesis.

Starting from Chapter 3, we actually begin to apply the concept introduced in Chapter 2 to the synonym acquisition task. To evaluate the performance, we propose the use of two similarity measures, i.e., average precision (AP) and correlation coefficient (CC), which both use existing thesauri such as WordNet. As the preprocessing of experiments, we firstly investigate the effect of frequency cut-off to word and context types, which has been used in all the experiments described in this thesis. Then we compare the similarity measures and weighting functions introduced in Chapter 3.

In Chapter 4, we introduce two latent semantic models, Latent Semantic Indexing (LSI) and Probabilistic Latent Semantic Indexing, which were both originally proposed for information retrieval tasks but can actually be applied to any co-occurrence models between two domains. We applied LSI and PLSI models to synonym acquisition, and show whether they are effective compared to the baseline methods we deal with in Chapters 2 and 3. Some related topics, including the effect of the inverse temperature of PLSI and the number of latent classes are to be investigated as well.

In Chapter 5, the problem of context representation and extension is addressed. Firstly we explain how the normal dependency-based context fails to capture long-distance semantic relationship. We then show the formalization of dependency-based context (`dbc`), and conduct an experiment to show that the extension of `dbc` using indirect dependency achieves higher synonym acquisition performance. We also present four types of context representations of indirect dependency and compare their result.

In Chapter 6, we propose three schemes of context selection — category-, type-, and co-occurrence based selection, to reduce the computational cost when computing distributional similarity. For category-based selection, we set the contextual categories based on the grammatical relation for dependency-based context, and window length and shift amount for word-based context. For type- and co-occurrence based selection, we introduce four or five importance score measures each, and compare their effectiveness in reducing the computational cost while keeping the performance.

In Chapter 4, we propose two novel approaches to synonym acquisition. The first one is classification-based synonym acquisition, where context-based features called *distributional features* are constructed from extracted context. Because this approach enables the integration of context-based features and syntactic pattern-based features, it can benefit various techniques from machine learning field. The second approach is based on information-theoretic metric learning, where an extension of Euclid distance called Mahalanobis distance is learned from the training sets of similar/dissimilar pairs of words. We show how these two approaches outperform the conventional distributional similarity-based approaches.

In Chapter 8, we conclude this thesis, wrapping up the findings and results obtained from each chapter. We also show our possible future direction of this study.

Chapter 2

Distributional Similarity

2.1 Introduction

Now assume that we are given these sentences and asked to "guess" the meaning of the unknown word *tezgüino* (taken from [Lin, 1998a]):

> A bottle of *tezgüino* is on the table.
> Everyone likes tezgüino.
> Tezgüino makes you drunk.
> We make tezgüino out of corn.

Actually, although we do not even know whether any words like *tezgüino* exist in this world[1], we can easily guess that the word *tezgüino* would, if any, mean some sort of alcoholic drink, maybe some sort of corn beer. This is because we can also reasonably say:

> A bottle of *corn beer* is on the table.
> Everyone likes *corn beer*.
> *Corn beer* makes you drunk.
> We make *corn beer* out of corn.

The reason why we assumed that these two words, *tezgüino* and *corn beer* have similar meanings must be that they have very similar *context*. The underlying assumption here is the so-called *distributional hypothesis* [Harris and (ed.), 1985], which states that semantically similar words share similar contexts. It means, in other words, the more similar the contexts of words are, the more likely the words are semantically related, and similarity can be obtained by comparing the contexts the target words have. The similarity obtained in this way is called *distributional similarity*, which is also referred as *second-order co-occurrence* in some cases [Picard, 1999].

Let us define two important concepts throughout this thesis: *word* and *context*. A *word*, or more precisely, a *word token*, is a unit of meaning accompanying context, which appears in a sentence in a corpus. After analyzing the corpus, a sentence is

[1] Actually the word *tezgüino* does exist, also spelled as *tesguino*, which is a Mexican corn beer.

considered a list of words, and the corpus is a list of sentences. On the other hand, a *word type* is the class of the words which have the same *type*, in most cases the same base form. Here's an example — a word type `company` can occur many times in a corpus, each of which is a single occurrence of the word type. One of the word tokens could be `company` in a sentence, and another could be `companies` in another sentence. The relationship between a word type and its word tokens is similar to that of a class and its instances (objects) in object-oriented programming (OOP) languages. We do not make this distinction between word tokens and types when it is obvious from the context.

A *context* (countable) is a piece of information associated with a particular word token. For example, in a phrase "a big company," the word token "company" is associated with a context "modified by an adjective *big*." Another context associated with the word token is "has an article *a*," and there can be a number of contexts associated with a single word token, depending on how contexts are defined. On the other hand, a *context type* is the class of the contexts which have the same type. For example, in another phrase "a populated city," the word token "city" also has a context "has an article *a*," which belongs to the same context type as the one *company* has in the above example. Using the OOP analogy again, the relation between a word token and its contexts is similar to an instance (object) and its properties. Yet another important concept is a *context set* of a word type, which is the set of all the contexts that all the instances of the word type have in a corpus. For example, given a business-related corpus, the context set of a word type `company` could contain "has an article *a*," "subject of a verb *hire*," "object of a verb *acquire*," and so forth.

Now that these concepts are defined, distributional similarity is more precisely defined as the similarity between word types measured by the commonality of their context sets. Context sets are often defined as vectors in an M-dimensional real vectors space, where the vector elements are usually the frequencies of the corresponding context type, and M is the total number of the extract context types. This model is called vector space model (VSM), and the constructed real vectors space is often called *semantic space*. The distributional similarity between word types is the similarity between the vectors corresponding to their context sets. The details of this formalization is given in the following sections.

Most of the acquisition methods can be regarded as the combination of all or some of these three steps: (1) context extraction, (2) similarity calculation, and (3) word clustering. Firstly, the methods extract useful context sets of words to characterize the target word types from large corpora. The context to be extracted varies from very naive ones such as surrounding words captured by a window to sophisticated ones such as dependency structure, as we showed above. Regardless of contexts in use, the key assumption on acquiring word similarity is the above-mentioned distributional hypothesis.

In the second step, *similarity*, which is equivalently referred as *semantic relatedness* in this thesis, is calculated based on the commonality of words' context sets. Various similarity measures and weight functions have been proposed for this purpose, which roughly fall into two mainstreams: vector- and probability-based similarities. The former includes the traditional tf.idf weighting and cosine similarity. The latter is

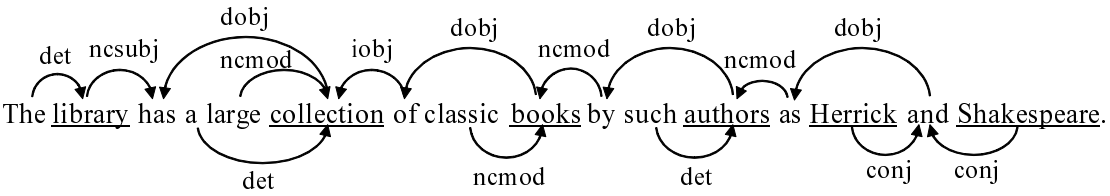

Figure 2.1: Dependency structure of the example sentence

represented by Kullbuck-Leibler (KL) divergence and Jensen-Shannon (JS) distance. These similarity measures and weight functions have been well compared and discussed so far [Chung and Lee, 2001, Curran and Moens, 2002a]. We will also discuss them in this chapter.

As the third step, word clustering is optionally conducted based on the similarity or distance calculated in the previous step to create clusters of semantically related words. Again, a wide variety of clustering methods are proposed [Pereira et al., 1993, Sakaki et al., 2007], although this issue is not within the scope of this thesis.

2.2 Context Extraction

Given a sentence, how to define contexts associated with a certain word token is an open issue. We use the following example for illustration purposes:

> *The library has a large collection of classic books by such authors as Herrick and Shakespeare.*

The contexts of the word token "collection" include "modified by an adjective *large*," "has an article *a*," "direct object of a verb *has*," and so on. We could also define contexts such as "the next word is *of*," "located at the 6th position in a sentence," or even "the sentence starts with a letter *T*," which might be less effectiveness for characterizing the meaning of the word than the previous three contexts. Indeed, various types of contextual information have been proposed for distributional similarity computation, including surrounding words [Curran and Moens, 2002b, Lowe and McDonald, 2000], dependency or case structure [Hindle, 1990, Ruge, 1997, Lin, 1998a], and dependency path [Lin and Pantel, 2001, Pado and Lapata, 2007]. The discussion on which contextual information to use is detailed in Chapters 5 and 6.

Because the modeling of the contextual information is not the focus here, in this chapter we simply adopt dependency structure as the contexts of words since it is the most widely used and it is shown to be well-performing contextual information in the past studies. We call this context *dependency-based context* in this thesis. Note that the approach or discussion described in this chapter is not limited to specific contextual information of choice. The detailed introduction about other contextual information is found in Section 1.2.

Throughout this thesis the sophisticated parser RASP Toolkit 2 [Briscoe et al., 2006] is utilized to extract this kind of word relations. When applied to the above example

sentence, RASP outputs the extracted dependency structure as n-ary relations as follows, whose graphical representation is shown in Figure 2.1:

```
(ncsubj have library _)
(dobj have collection)
(det collection a)
(ncmod _ collection large)
(iobj collection of)
(dobj of book)
(ncmod _ book by)
(dobj by author)
(det author such)
(ncmod _ author as)
...
```

These relations specify which word tokens have which kind of dependency relation. For example, the first tuple `(ncsubj have library _)` specifies that the word token `library` is the subject of the verb `have`. The prefix `nc` stands for "non clausal" in this tag-set.

While the RASP outputs are n-ary relations in general, what we need here is *co-occurrences* of word tokens and their contexts, so we extract the set of co-occurrences of stemmed words and their contexts by taking out the target word from the relation and replacing the slot by an asterisk "`*`":

```
library     - (ncsubj have * _)
library     - (det * The)
collection  - (dobj have *)
collection  - (det * a)
collection  - (ncmod _ * large)
collection  - (iobj * of)
book        - (dobj of *)
book        - (ncmod _ * by)
book        - (ncmod _ * classic)
author      - (dobj by *)
author      - (det * such)
...
```

In this way, all the contexts of every word token appearing in the sentence are obtained. For example, the contexts of `library` will be `(ncsubj have * _)` and `(det * The)`. This process is applied to all the tokens in a corpus, and by summing up all the co-occurrences, we can obtain the frequency of a word type w having a context type c in its context set. We denote this frequency as $n(w, c)$ in the following. Using this co-occurrence count, we can denote the context set $C(w)$ of a word type w as

$$C(w) = \{c | n(w, c) > 0\}. \tag{2.1}$$

Also, a word vector $\mathbf{w}$ of w is defined as

$$\mathbf{w} = [n(w, c_1) \quad n(w, c_2) \quad \ldots \quad n(w, c_M)]^T, \tag{2.2}$$

where M is the number of all the context types appearing in a corpus.

2.3 Similarity Measures

Once context sets are formally defined, distributional similarity is defined based on how common the context sets of two word types are:

$$\mathrm{sim}(w_1, w_2) = \mathrm{comm}(C(w_1), C(w_2)) \tag{2.3}$$

or how similar the word vectors are:

$$\mathrm{sim}(w_1, w_2) = \mathrm{sim}(\mathbf{w}_1, \mathbf{w}_2), \tag{2.4}$$

or even how large the distance between the vectors is:

$$\mathrm{dist}(w_1, w_2) = \mathrm{dist}(\mathbf{w}_1, \mathbf{w}_2). \tag{2.5}$$

Note that we need to convert distance, i.e., dissimilarity, to similarity in the last case, unless we are only dealing with *ranking* of related words. When ranking related words, they are sorted in a descending order of similarity or an ascending order of distance. When we need the similarity value from the distance, we can convert as:

$$\mathrm{sim}(w_1, w_2) = \exp\{-\lambda \mathrm{dist}(w_1, w_2)\}, \tag{2.6}$$

where λ is an appropriate scaling factor. This conversion is used in [Mochihashi and Matsumoto, 2002]. Alternatively, we can simply convert as:

$$\mathrm{sim}(w_1, w_2) = C - \mathrm{dist}(w_1, w_2), \tag{2.7}$$

where C is a large enough constant. This conversion is used in [Lee, 1999]. We use Equation (2.6) throughout this thesis, with $\lambda = 0.1$.

There can be a number of ways to define this commonality or distance of two context sets, and many have been actually proposed and compared in previous work [Lin, 1998b, Lee, 1999, Lee, 2001, Curran and Moens, 2002a, Weeds and Weir, 2003, Weeds et al., 2004]. However, thorough and comprehensive comparison of these measures is not the focus of this thesis. We choose some of the popular ones and use them as baseline of the proposed methods in this thesis. In the following sections, we introduce the measures we chose, which can be roughly classified into two categories — similarity-based and distance-based methods.

2.3.1 Similarity-based Measures

The similarity measures we employ to calculate the inter-word similarity are listed below: `cosine`, `jaccard-s`, `jaccard-v`, and `linsim`. In the following, $\mathbf{w}_1$ and $\mathbf{w}_2$ are word vectors of w_1 and w_2, and $\mathrm{wgt}(w, c)$ is the weight value assigned to the co-occurrence of a word type w and a context type c. How to assign this value is later discussed in Section 2.4.

Cosine Similarity (`cosine`) One of the most commonly used similarity functions, especially for information retrieval, along with the vector space model is cosine similarity:

$$\text{sim}_{\text{cosine}}(w_1, w_2) = \frac{\mathbf{w}_1 \cdot \mathbf{w}_2}{||\mathbf{w}_1|| \cdot ||\mathbf{w}_2||} = \frac{\sum_{c \in C(w_1) \cap C(w_2)} \text{wgt}(w_1, c) \times \text{wgt}(w_2, c)}{\sqrt{\sum_{c \in C(w_1)} \text{wgt}(w_1, c)^2 \times \sum_{c \in C(w_2)} \text{wgt}(w_2, c)^2}} \quad (2.8)$$

which is basically the inner product of two vectors $\mathbf{w}_1$ and $\mathbf{w}_2$, normalized by the vector lengths.

Set-based Jaccard Coefficient (`jaccard-s`) Jaccard coefficient is a similarity measure originally used to compute the commonality of two sets X and Y, and defined as:

$$J(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}. \quad (2.9)$$

In this naive form, it only uses the number of elements in sets, but here the measure is extended and used for distributional similarity, and the weights are also taken into consideration as the following:

$$\text{sim}_{\text{jaccard-s}}(w_1, w_2) = \frac{\sum_{c \in C(w_1) \cap C(w_2)} \min(\text{wgt}(w_1, c), \text{wgt}(w_2, c))}{\sum_{c \in C(w_1) \cup C(w_2)} \max(\text{wgt}(w_1, c), \text{wgt}(w_2, c))}. \quad (2.10)$$

Vector-based Jaccard Coefficient (`jaccard-v`) There is another extension of Jaccard coefficient which is based on the commonality of two vectors:

$$\begin{aligned} &\text{sim}_{\text{jaccard-v}}(w_1, w_2) \\ &= \frac{\mathbf{w}_1 \cdot \mathbf{w}_2}{||\mathbf{w}_1||^2 + ||\mathbf{w}_2||^2 - \mathbf{w}_1 \cdot \mathbf{w}_2} \\ &= \frac{\sum_{c \in C(w_1) \cap C(w_2)} \text{wgt}(w_1, c) \times \text{wgt}(w_2, c)}{\sum_{c \in C(w_1)} \text{wgt}(w_1, c)^2 + \sum_{c \in C(w_2)} \text{wgt}(w_2, c)^2 - \sum_{c \in C(w_1) \cap C(w_2)} \text{wgt}(w_1, c) \times \text{wgt}(w_2, c)}. \end{aligned}$$

This extension is also called *Tanimoto coefficient* [Tanimoto, 1957].

Note that we still have *Dice coefficient*, which is also widely-adopted in many similarity-based tasks. However, we omitted it from choice this time, because it is shown that Dice and Jaccard coefficient are monotonic in each other [Weeds et al., 2004], yielding exactly the same results.

Lin's Similarity Measure (`linsim`) Lin's similarity measure [Lin, 1998a] is yet another similarity measure which has been widely used and compared for distributional similarity tasks [Geffet and Dagan, 2004, Weeds et al., 2004], and [Curran and Moens, 2002a]. This similarity measure is derived from a formal and general definition of similarity described in [Lin, 1998b], which is given by:

$$\mathrm{sim}_{\mathrm{linsim}}(w_1, w_2) = \frac{\sum_{c \in C(w_1) \cap C(w_2)} \mathrm{wgt}(w_1, c) + \mathrm{wgt}(w_2, c)}{\sum_{c \in C(w_1)} \mathrm{wgt}(w_1, c) + \sum_{c \in C(w_2)} \mathrm{wgt}(w_2, c)}. \tag{2.11}$$

2.3.2 Distance-based Measures

There is a distinct class of metrics which capture word relatedness, i.e., distance-based measures. These measures computes the distances, or unrelatedness, between two word types, based on the corresponding word vectors or *context distribution* of word types.

Firstly, we introduce two distance-based measures, `euclid` and `manhattan`, both of which are defined on two word vectors.

Euclid distance (`euclid`) Euclid distance, or L2 distance, is the most popular distance metric between two Euclidean vectors, which is defined as:

$$\mathrm{dist}_{\mathrm{euclid}}(w_1, w_2) = ||\mathbf{w}_1 - \mathbf{w}_2|| = \sqrt{\sum_c \left(\mathrm{wgt}(w_1, c) - \mathrm{wgt}(w_2, c)\right)^2} \tag{2.12}$$

Note that if word vectors are L2-normalized, i.e., normalized so that their L2-norm is equal 1:

$$\mathbf{w}' = \frac{\mathbf{w}}{||\mathbf{w}||}, \tag{2.13}$$

then

$$\mathrm{sim}_{\mathrm{cosine}}(w_1', w_2') = \mathbf{w}_1 \cdot \mathbf{w}_2, \tag{2.14}$$

$$\mathrm{dist}_{\mathrm{euclid}}(w_1', w_2') = 2(1 - \mathbf{w}_1 \cdot \mathbf{w}_2). \tag{2.15}$$

Then, cosine similarity and Euclid distance are inversely monotonic to each other. In this case, it is redundant to compare L2-normalized Euclid distance with cosine similarity, so we do not L2-normalize vectors when using Euclid distance.

Manhattan distance (`manhattan`) Manhattan distance, or L1 distance, is another distance metric between two Euclidean vectors, which is given by:

$$\mathrm{dist}_{\mathrm{manhattan}}(w_1, w_2) = \sum_c |\mathrm{wgt}(w_1, c) - \mathrm{wgt}(w_2, c)|. \tag{2.16}$$

Secondly, we introduce other two distance-based measures, `js` and `sd99`, both of which are defined on two context distributions. Here the term *context distribution* means a probability distribution over the set of all context types, which is created

by converting co-occurrence weights $\mathrm{wgt}(w_i, c_j)$. Specifically, the context distribution $p(X; w_i)$ of a word w_i with a random variable X is defined as following:

$$p(X = j; w_i) = \frac{1}{z(w_i)} \left[\mathrm{wgt}(w_i, c_j) - s(w_i)\right], \tag{2.17}$$

where $s(w_i)$ is to shift the weight values so that all the probabilities are greater or equal to 0, which is defined as

$$s(w_i) = \min(0, \min_c \mathrm{wgt}(w_i, c)). \tag{2.18}$$

Here $z(w_i)$ is the normalization factor which is defined such that $\sum_j p(X = j; w_i) = 1$.

Perhaps the most popular inter-distribution distance metric is Kullback-Leibler (KL) divergence [Lin, 1991]:

$$KL(p(w_1)||p(w_2)) = \sum_j p(j; w_1) \log \left[p(j; w_1)/p(j; w_2)\right], \tag{2.19}$$

which is widely used for measuring the distance between two probability distributions. However, it has some disadvantages such as asymmetricity and zero frequency problem, that is, if there exists j such that $p(j; w_1) \neq 0$ and $p(j; w_2) = 0$, the distance is not defined.

Jensen Shannon divergence (`js`) Jensen Shannon (JS) divergence [Lin, 1991] is a metric which we can consider as a symmetrized KL divergence:

$$\mathrm{dist}_{\mathrm{js}}(w_1, w_2) = KL(p(w_1)||m(p(w_1), p(w_2))) + KL(p(w_2)||m(p(w_1), p(w_2))), \tag{2.20}$$

where $m(p(w_1), p(w_2))$ is the *average* of two distributions $p(w_1)$ and $p(w_2)$:

$$m(X = j; p(w_1), p(w_2)) = \frac{1}{2} \left[p(X = j; w_1) + p(X = j; w_2)\right]. \tag{2.21}$$

JS divergence has some favorable properties: it is bound [Lin, 1991] and does not cause the zero frequency problem.

Skew divergence (`sd`) Skew divergence (SD), which is another metric receiving attention recently, has solved the zero frequency problem by introducing a parameter α and mixing two distributions:

$$\mathrm{dist}_{\mathrm{sd}}(w_1, w_2; \alpha) = KL(p(w_1) \mid\mid \alpha p(w_2) + (1 - \alpha) p(w_1)). \tag{2.22}$$

It has shown that Skew Divergence achieves better performance than the other measures [Lee, 2001]. In a preliminary experiment, we investigated how this parameter α affects the overall performance of distributional similarity, and chose $\alpha = 0.99$, which achieved the highest performance.

2.4 Weighting Functions

Just using the raw co-occurrence frequency can be problematic in some cases. Consider a case where some high-frequency context types co-occur with most of the word types. We do have these context types, e.g., "has an article *the*," "object of a verb *have*" and "modified by a possessive pronoun *his*," and two word sharing these high-frequency context types are not necessarily similar. On the other hand, two words having such context types as "subject of a verb *hire*" and "object of a verb *acquire*" in common tend to be similar, and these relatively infrequent context types give much more confidence in similarity calculation than frequent ones do.

The same situation can be seen for word types. Consider a case where we have a high-frequency word type w_h and a low-frequency word type w_l, and a context type c which co-occurs once with both word types. In this case, although $n(w_h, c) = n(w_l, c) = 1$, the co-occurrence (w_l, c) has more contribution to the characterization of the word w_l than the co-occurrence (w_h, c) has to w_h, thus the former should be given a higher weight. This discussion leads us to believe that each co-occurrence (w, c) should be given a weight $\mathrm{wgt}(w, c)$ based on the frequency of both word and context types, so that it reflects the contribution which each co-occurrence has to the characterization of words.

Indeed, there have been many weighting functions proposed so far [Curran and Moens, 2002a, Weeds et al., 2004]. Again, comprehensive comparison of weighting functions is not the focus of this thesis, thus we pay attention to these four weight functions listed below: `tf`, `tfidf`, `pmi`, and `ttest`.

Raw Frequency (`tf`) The raw co-occurrence frequency of word and context type is used as it is:

$$\mathrm{wgt}(w, c) = n(w, c). \tag{2.23}$$

Note that, although the terms *tf* and *idf* are based on information retrieval task setting, the modeling is essentially the same as distributional similarity and we still use the terms *tf* and *idf* in this thesis. Documents and terms in IR correspond to word and context types in distributional similarity, respectively.

tf.idf Weighting (`tfidf`) The raw frequency `tf` is weighted by idf:

$$\mathrm{wgt}(w, c) = n(w, c) \cdot \mathrm{idf}(c), \quad \mathrm{idf}(c) = \log \frac{n}{\mathrm{df}(c)}, \tag{2.24}$$

where $\mathrm{df}(c)$ is the number of word types with which the context type c co-occurs, i.e., $\mathrm{df}(c) = |\{w | n(w, c) > 0\}|$. In the experiments, the idf values are normalized so that the maximum equals to 1.

Pointwise Mutual Information (`pmi`) The amount of information that occurrences of w and c have in common is defined as their pointwise mutual information

(`pmi`):

$$\text{wgt}(w, c) = \text{PMI}(w, c) = \log \frac{P(w, c)}{P(w)P(c)}, \tag{2.25}$$

where $P(w, c)$ is the probability of the word w co-occurring with the context c. This probability is empirically calculated as

$$P(w, c) = n(w, c) / \sum_{w', c'} n(w', c') \tag{2.26}$$

using the co-occurrence frequency $n(w, c)$ obtained from corpora.

There are two things to note here: when $n(w, c) = 0$ and PMI cannot be defined, then we simply define $\text{wgt}(w, c) = 0$. Also, because it has been shown [Curran and Moens, 2002a] that negative PMI values worsen the distributional similarity performance, we bound the PMI so that $\text{wgt}(w, c) = 0$ if $\text{PMI}(w, c) < 0$.

t-statistic (`ttest`) t-statistic, which is normally used to test whether the appearances of a word w and a context c are independent, can also be used as a weight function:

$$\text{wgt}(w, c) = \frac{P(w, c) - P(w)P(c)}{\sqrt{P(w)P(c)}}. \tag{2.27}$$

Whereas t-statistic is usually employed to discover collocations from corpora, Curran and Moens [Curran and Moens, 2002a] proposed using it as a weight function and showed that it achieved higher performance compared to the others.

Chapter 3

Preliminary Experiments

In this chapter, we firstly describe how we are going to evaluate the performance of synonym acquisition methods throughout the entire thesis. We propose two evaluation methods, average precision (AP) and correlation coefficient (CC), which both rely on the gold standard obtained from the exiting thesauri such as WordNet.

Then we conduct experiments regarding the following topics: frequency cut-off, similarity measures, and weighting functions, which are all important aspects of the distributional similarity-based methods.

3.1 Evaluation of Distributional Similarity

Throughout this thesis, we employed two evaluation methods to evaluate the computed distributional similarity. The first one, average precision (AP), is based on the task of synonym acquisition and it evaluates the precision of acquired synonyms. The second one, correlation coefficient (CC), directly evaluates the quality of assigned similarity values by using the gold standard, namely, the similarity values obtained from WordNet.

We introduced those two measures to evaluate the effectiveness of distributional similarity from different viewpoints. AP evaluates the precision of acquired synonyms, i.e., how accurately synonyms are extracted, while CC evaluates how accurately the similarity is assigned to each word pair. For example, because the AP value will be high as long as the synonyms are accurately extracted, it cannot directly evaluate how similarity is assigned to non-synonym pairs. In short, AP evaluates the quality of *synonyms*, and is more important in some tasks such as synonym acquisition, while in other tasks, such as similarity-based word-count smoothing, CC is more direct evaluation because it evaluates the quality of *similarity*. By using them both, context modeling and selection methods described in this thesis in various tasks can become more clear.

3.1.1 Average Precision

The first evaluation measure, *average precision* (AP), is a common evaluation scheme mainly adopted for information retrieval, which evaluates how accurately the method under evaluation is able to extract synonyms.

To calculate AP, we firstly prepare a set of *query words*, for which synonyms are obtained to evaluate the precision. We adopted the Longman Defining Vocabulary (LDV) [1], originally consisted of 2,194 entries, as the candidate set of query words. Secondly, for each entry in LDV, three existing thesauri are consulted: Roget's Thesaurus [Editors of the American Heritage Dictionary, 1995], Collins COBUILD Thesaurus [Collins, 2002], and WordNet [Fellbaum, 1998]. The union of synonyms obtained when an LDV word is looked up as a noun is used as the *reference set*, except for words marked as "idiom," "informal," "slang," and phrases comprised of two or more words. Notice that most of the LDV entries have more than two senses in WordNet, in which case the union of synonyms found in all synsets of the entry was used as the reference set, since the synonym acquisition model described in this thesis is not capable of treating multiple senses of a single word. The LDV words for which no noun synonyms are found in any of the reference thesauri are omitted.

In general, the problem of such polysemies should not be treated lightly in synonym acquisition tasks. However, in context selection and modeling methods we address in this thesis, we believe that the effectiveness of (extended) context types can still be universally determined, whether or not the synonym model is polysemy-aware, and the effect of polysemies is limited in the experiments described in this thesis.

Finally, from the remaining 771 LDV words, 100 query words are randomly chosen, which are listed in Appendix. For each of them, the eleven precision values at 0%, 10%, ..., and 100% recall levels are averaged to calculate the final AP value.

3.1.2 Correlation Coefficient

The second evaluation measure is *correlation coefficient* (CC) between the target similarity and the *reference similarity*, i.e., the gold standard of similarity for word pairs. The CC value is calculated so that it becomes larger when the obtained similarities are more precise approximation of the reference similarities, which are pre-computed using the WordNet structure.

As the reference similarity, we used Lin's similarity measure [Lin, 1998b]:

$$\mathrm{sim}_c(x_1, x_2) = \frac{2 \log P(C_0)}{\log P(C_1) + \log P(C_2)}, \tag{3.1}$$

where C_1 and C_2 are WordNet classes corresponding to words w_1 and w_2, respectively. These classes have $x_1 \in C_1$ and $x_2 \in C_2$ as individual senses, and the most specific class that subsumes both C_1 and C_2 is denoted as C_0. The word similarity between the word w_1 and w_2 is calculated by taking the maximum class similarity value of all

[1] http://www.cs.utexas.edu/users/kbarker/working_notes/ldoce-vocab.html

the combinations of senses as:

$$\mathrm{sim}_w(w_1, w_2) = \max_{x_1 \in C_1, x_2 \in C_2} \mathrm{sim}_c(x_1, x_2). \tag{3.2}$$

We have to note that a variety of similarity/distance measures are proposed to obtain word relatedness based on WordNet structure [Budanitsky and Hirst, 2006]. However, we conducted a preliminary experiment and verified that the different values of CC calculated using various calculation techniques including the one mentioned above and [Jiang and Conrath, 1997, Hirst and St-Onge, 1998], all of which can be calculated utilizing Wordnet::Similarity package [Pedersen et al., 2004], are highly correlated and the choice does not essentially affect the overall evaluation.

The value of CC is calculated as the correlation coefficient of reference similarities $\vec{r} = (r_1, r_2, ..., r_n)$ and target similarities $\vec{s} = (s_1, s_2, ..., s_n)$ over the word pairs in sample set P_s, which the set of 10,000 randomly created pairs from LDV.

3.2 Experimental Settings

We extracted contextual information from the corpus: the "story"-type documents of New York Times articles (1994) in English Gigaword[2], consisting of 45,830 documents and approx. 30 million words. We used the NYT corpus in other experiments in this thesis. Because this corpus is a collection of broad topics of sentences, and it is not limited to business topics, we can reasonably say that the findings of this thesis are general enough and not too much dependent on the corpus.

As the extraction of accurate and comprehensive syntactic relations is in itself a difficult task, the sophisticated parser RASP Toolkit ver. 2 (RASP2) [Briscoe et al., 2006] was utilized to extract dependency relations. RASP2 analyzes input sentences and provides a wide variety of grammatical information such as POS tags, dependency structure, and parsed trees as output, among which we employed the dependency structure called grammatical relations (GRs), which we used to create the co-occurrence frequency $n(w, c)$. According to [Briscoe et al., 2006], the RASP2 system achieves a micro-averaged F_1 score of 76.3% for relation assignment. Even when the parser makes some errors, we suppose the influence would not be serious — the wrongly assigned relations act as less precise contexts, still working better than the naive surrounding word-based context.

Since our purpose here is the automatic extraction of synonymous nouns, only the contexts for nouns are extracted. To distinguish nouns, using POS tags annotated by RASP2, any words with POS tags APP, ND, NN, NP, PN, and PP, were labeled as nouns.

3.3 Frequency Cut-off

One of the important factors when applying distributional similarity is *cut-off frequencies*, i.e., thresholds on occurrence frequency to filter out words or contexts with low

[2]http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2003T05

frequency to reduce computational cost. More specifically, we set two frequencies, θ_w and θ_c, which are to remove low-frequency word and context types, respectively, and any words w such that $\sum_c n(w, c) < \theta_w$ and any contexts c such that $\sum_w n(w, c) < \theta_c$ were removed from the co-occurrence data. Special attention is required to fix the values of θ_w and θ_c because they can directly affect the size of constructed semantic vector spaces and/or the performance of distributional similarity.

The first one, the word cut-off frequency θ_w, is relatively easier to handle, because it affects the number of word types extracted from corpora almost exclusively. Assuming the Zipf's law, the number of the remaining word types is inversely proportional to θ_w, which we confirmed through a preliminary experiment. When fixing θ_w, what we should consider is the trade-off between the remaining word types and the computational cost — the threshold must be high enough to make it computationally practical, and low enough to keep as much proportion of the original word types as possible.

The second one, the context cut-off frequency θ_c, is less easy to handle, because it affects both the semantic space size and the performance (AP and CC). Although the number of the remaining context types can be easily estimated using Zipf's law, similarly to θ_w, we have no other ways to know the performance change but to empirically investigate it. In the experiment, we investigate their effects separately, and show how they affect the performance.

3.4 Results

In this section, we conduct two experiments — one is to investigate the effect of the frequency cut-off we described in the previous section, and the other is to compare the performances of similarity measures and weighting functions described in Sections 2.3 and 2.4.

3.4.1 Effect of Frequency Cut-off

Firstly, the performance was evaluated when the word cut-off θ_w is increased from 2 to 30 with a step of 2, while fixing the context cut-off θ_c. We used `jaccard-s` as similarity measure and `pmi` as weighting function in this experiment, but the trend we observed here is not particular to these settings we used. Figure 3.1 shows the change of performance (AP and CC), where three lines are drawn, each corresponding to the average of the θ_c range ($2 \leq \theta_c \leq 10$, $12 \leq \theta_c \leq 20$, and $22 \leq \theta_c \leq 30$). It shows clear increase of AP when θ_w is increased. This result is easy to interpret, because setting θ_w helps to make low-frequency, noisy words disappear from the synonym ranking and to improve the performance. Although this trend is independent of the θ_c range, the boosting effect of θ_w dissipates as it gets larger.

On the other hand, the result shows that θ_w hardly affects CC, and it only wavers within 10% range. This outcome is also explicable, because technically θ_w is not supposed to affect CC at all (remember it only measures the quality of assigned similarities, not the ranking). In other words, θ_w only removes low-frequency word

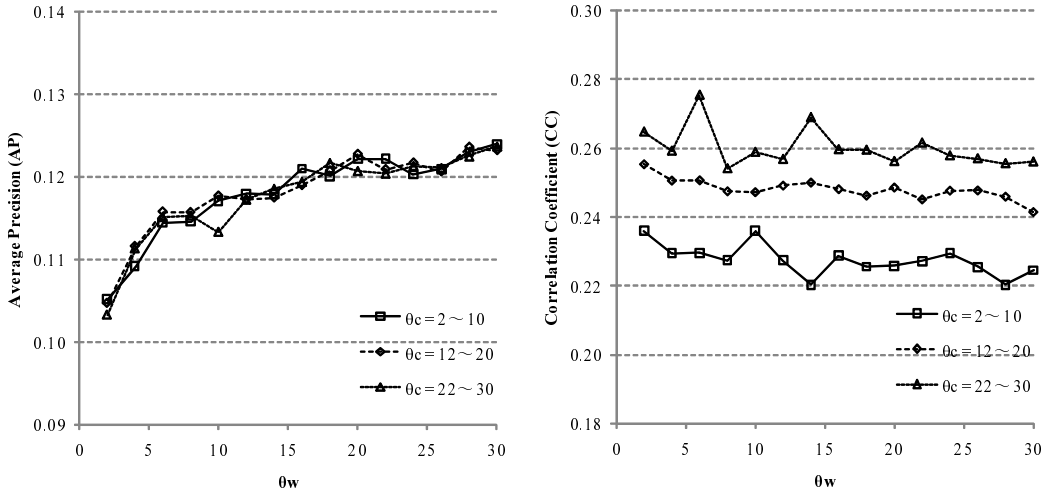

Figure 3.1: Effect of word cut-off θ_w

types, and this has nothing to do with the assigned similarity of the remaining word types. The reason why CC wavers is the weighting, namely `pmi` in this case, is actually affected by the removed word types, although its effect can be almost ignored.

Next, we investigated the effect of the context cut-off θ_c, by similarly increasing it from 2 to 30 with a step of 2, while fixing the word cut-off θ_w. Figure 3.2 shows how AP and CC change, where three lines for the average of $2 \leq \theta_w \leq 10$, $12 \leq \theta_w \leq 20$, and $22 \leq \theta_w \leq 30$ are drawn. A surprising result is seen for AP, because it turns out that θ_c does not affect AP very much. This result suggests that synonym ranking is not very susceptible to context selection.

On the contrary, the result clearly shows that CC gets higher while θ_c is increased. This result is more intuitive, because we suppose that increasing θ_c removes infrequent, thus low-quality, context types, and this would increase the accuracy of the assigned similarity values. Although these results of AP and CC are somewhat contradictly, they can be explained based on the characteristics of AP and CC and context commonality. Detailed discussion is provided in Chapter 6.

Considering this result, we adopted $\theta_w = \theta_c = 20$ in the following experiments, which is the point where the increase of performance seems to reach a plateau. The numbers of the remaining word and context types are 15,110 and 34,570, respectively.

3.4.2 Similarity Measures and Weighting Functions

In this section, we firstly compared all the similarity measures we introduced — `cosine`, `jaccard-s`, `jaccard-v`, `linsim`, `euclid`, `manhattan`, `js`, and `sd99`. While comparing these measures, the weighting function was fixed to `pmi`.

The result is shown in Table 3.1. Top three performance values (both AP and CC) are emphasized in bold. We can observe that best measures are `cosine` and `js`, and next come `jaccard-v` and `linsim`. The AP values are higher for some measures

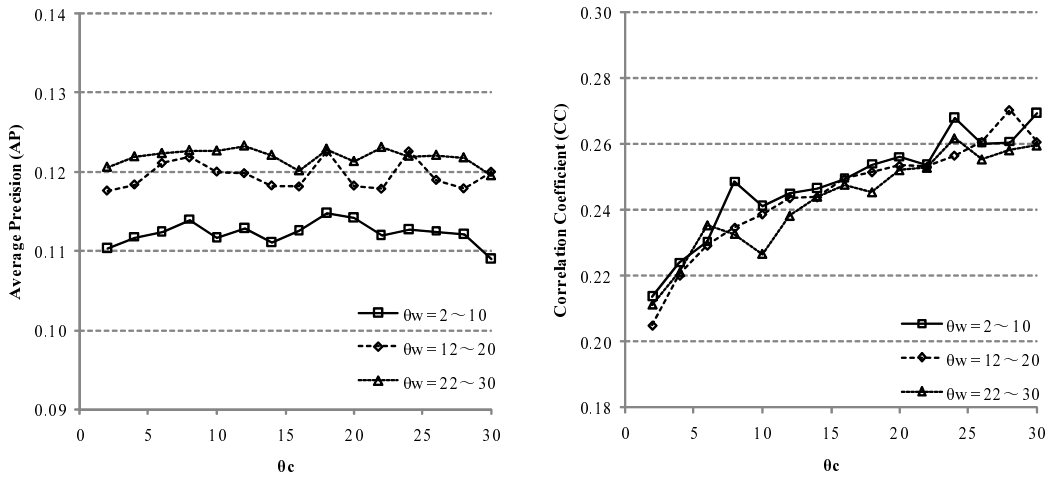

Figure 3.2: Effect of context cut-off θ_c

Table 3.1: Comparison of similarity measures

Similarity Measure	Average Precision (AP)	Correlation Coefficient (CC)
`cosine`	**0.1300**	**0.3101**
`jaccard-s`	0.1233	0.2567
`jaccard-v`	**0.1309**	0.2573
`linsim`	0.1258	**0.3296**
`euclid`	0.0022	0.0369
`manhattan`	0.0040	0.0433
`js`	**0.1297**	**0.3236**
`sd99`	0.0967	0.2874

while the CC are higher for some others. However, it is a general tendency that AP and CC correlate with each other quite well. On the other hand, the vector-based distance measures `euclid` and `manhttan` are poorly performing. This may be because they are not L2-normalized, and the magnitude of a vector affects the obtained final distances much more than the length does. By normalizing and mapping vectors onto hypersphere with the radius of 1 could solve this problem. Indeed, `euclid` corresponds to the angle (cosine) between vectors after this operation as we discussed in Section 2.3. Considering that probability-based distance measures, namely `js` and `sd99` also involve some kind of normalization which converts vectors to context distributions, some form of normalization is indispensable when applying distance-based measures.

We then compared all the weighting functions we introduced — `tf`, `tfidf`, `pmi`, and `ttest`. While comparing these functions, the similarity measure was fixed to `cosine`, because it performed the best in the previous experiment.

The result is shown in Table 3.2. Top three performance values (both AP and CC) are emphasized in bold. The best performing function, as the result shows, is

Table 3.2: Comparison of weighting functions

Weighting Function	Average Precision (AP)	Correlation Coefficient (CC)
`tf`	0.0354	0.0314
`tfidf`	0.0576	0.0743
`pmi`	**0.1300**	**0.3101**
`ttest`	**0.0678**	**0.2175**

clearly `pmi`, and next comes `ttest`. On the other hand, `tf` performs poorly, even when it is weighted and `tfidf` is used. The difference between these two groups, i.e., `tf` and `idf` v.s. `pmi` and `ttest`, is that the latter incorporates the word-based normalization factor $P(w)$, which might be the cause of their performance superiority. An interesting fact is that the vectors will be normalized by the normalizing factors of similarity measures anyhow, e.g., $||\mathbf{w}||$ in the denominator of `cosine`. Still, it might be effective to incorporate a normalization factor as part of a weighting function.

3.5 Conclusion

In this chapter, we described the two evaluation measures we employed to evaluate synonym acquisition tasks, average precision (AP) and correlation coefficient (CC). These two measures are able to evaluate the task from two different points of view — acquired synonyms and calculated similarity. The difference will be more evident in Chapter 6.

We then conducted an experiment to investigate how the performance is affected by the cut-off frequencies, i.e, θ_w and θ_c. The experiment showed that these parameters actually worked to increase the performance in some cases, while reducing the word/context types greatly.

The second experiment compared the similarity measures and weighting functions we introduced in Sections 2.3 and 2.4. The result showed that `cosine`, `jaccard-v`, and `js` worked well, and none of the the similarity measures which do not incorporate some type of normalization achieved as good performance as the others. As for the weighting functions, the performance of `pmi` and `ttest` was the best, and this may be because they both incorporate word-based normalization factor $P(w)$, while the other two do not. The result described in this chapter will be the fundamental of all the following chapters.

Chapter 4

Modeling of Distributional Similarity

4.1 Introduction

In the last two chapters, we have only been paying attention to the vector space model (VSM), as one way to express the distributional hypothesis. Although this model is simple and effective, and it works quite effectively most of the time, it is not suitable for handling two important problems in lexical knowledge acquisition — synonymy (words with similar meaning) and polysemy (single words with multiple meanings). This is because all the context types, which correspond to dimensions in the VSM, are treated being independent, although actually they are not. A simple example shows this — consider target words "company" and "department" and assume that the former co-occurs with a context type "subject of a verb *employ*" and the latter with "subject of a verb *hire*." These two context types are obviously related — in fact they are almost synonymous, giving a strong evidence that these two target words "company" and "department" are related in that they are both something that can hire people. However, the simple VSM cannot capture this relationship, and these two context types would not give any evidence to the similarity of these two words.

To solve this issue, relevance or correlation among dimensions of a vector space model should be considered somehow. In other words, we have to discover the intrinsic components that contribute to characterization of word types. Several latent semantic models have been proposed so far, mainly for document indexing targeted at information retrieval. The most popular ones are Latent Semantic Indexing (LSI) [Deerwester et al., 1990] and Probabilistic LSI [Hofmann, 1999]. The former is strongly related to the principal component analysis (PCA) technique, which tries to find intrinsic components that contribute to the indexing of documents. The latter is a generative model where documents and terms are both generated via latent classes. The number of latent semantic classes is usually fewer than the original dimensionality, so using these methods achieves dimensionality compression effect.

In this chapter, we apply these latent semantic models to the synonym acquisition problem. As we showed in Chapter 2, the distributional similarity can be modeled

by word-context co-occurrence. This modeling is exactly the same as the document-term co-occurrence model used in information retrieval. Thus we expect that the latent semantics of words can be represented by applying these models, increasing the performance of synonym acquisition. In Section 4.2, we introduce the LSI model. The following Section 4.3 explains the PLSI model, along with its related topics, i.e., the aspect model, the EM algorithm, implementation, and the tempered EM algorithm. We then describe the experiment, where we compared the performance of the simple vector space model, LSI, and PLSI in Section 4.4. Section 4.5 concludes this chapter.

4.2 Latent Semantic Indexing (LSI)

Latent Semantic Indexing (LSI) is a latent semantic model which tackles the synonymy and polysemy issue, utilizing a technique similar to principle component analysis (PCA). In LSI, *word-context co-occurrence* matrix X

$$X = [\mathbf{w}_1\ \mathbf{w}_2\ ...\ \mathbf{w}_N], \tag{4.1}$$

is firstly constructed. This matrix is called *document-term* matrix in IR, consisting of all the word vectors as column vectors. This matrix X is then decomposed using singular value decomposition (SVD) as:

$$X = T_0 S_0 D_0^T. \tag{4.2}$$

Supposing $r = \mathrm{rank}(X)$, T_0 is $M \times r$ matrix of left singular vectors as column vectors, D_0^T is $r \times N$ matrix of right singular vectors as column vectors, and S_0 is $r \times r$ diagonal matrix of singular values. We call these matrices *term matrix*, *document matrix*, and *singular value matrix*, following the IR convention. Here we suppose that the singular values in S_0 are ordered in a descending order, thus the left and right singular vectors are ordered in a descending order of the corresponding singular values. The higher the corresponding singular value is, the higher the contribution of that component is.

In our model of distributional similarity, each row in the document matrix D_0^T corresponds to the recomposed vector of each word type w, with its element arranged in an order of importance. Therefore, keeping only $K(K < r)$ most important elements and removing the rest will produce dimensionality and noise reduction effect. The remaining elements have high contribution to word characterization, thus we can consider them as the latent semantics of the word types. The resultant document matrix is denoted D^T, and the same reduction procedure is applied to T_0 and S_0, producing T and S, respectively. The re-combination of T, S, and D yields the approximated matrix $\hat{X}$:

$$X \simeq \hat{X} = TSD^T. \tag{4.3}$$

The dimensionality reduction procedure for the document vectors is illustrated in Figure 4.1. LSI is reported to improve the information retrieval performance compared to conventional methods [Deerwester et al., 1990].

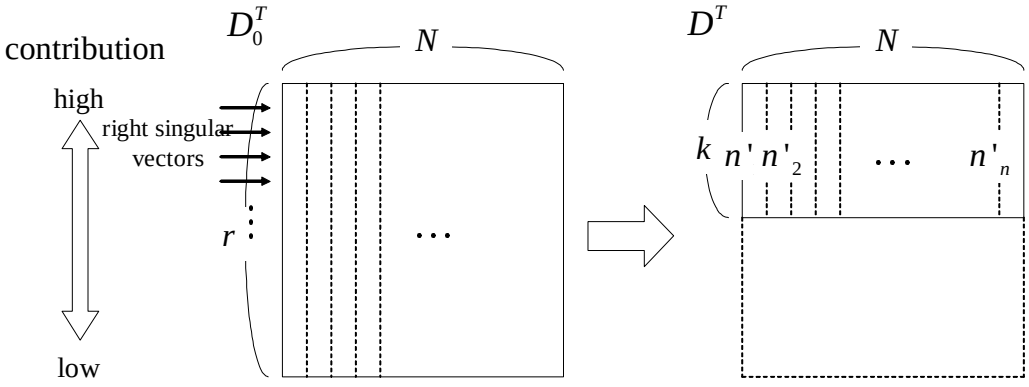

Figure 4.1: Dimensionality reduction procedure of document vectors

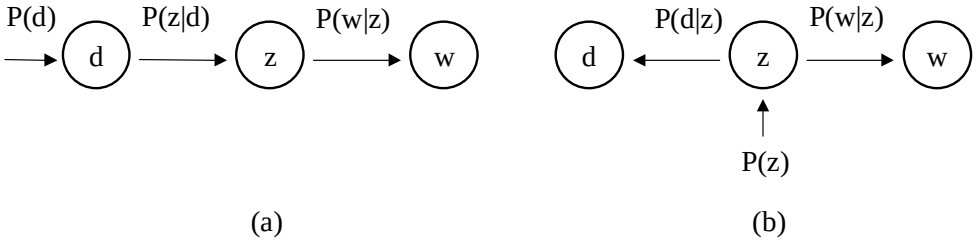

Figure 4.2: PLSI model: (a) asymmetric and (b) symmetric parameterization

The word vectors after the dimensionality reduction are denoted as $\mathbf{w}'_1, \mathbf{w}'_2, ..., \mathbf{w}'_N$, each of which has K dimensions. After this procedure, similarities between words can be calculated exactly as done for distributional similarity, using $\mathbf{w}'$ instead of $\mathbf{w}$.

4.3 Probabilistic Latent Semantic Indexing (PLSI)

4.3.1 Aspect model

We firstly describe the aspect model [Hofmann et al., 1999], which is the basis of the PLSI model. The aspect model itself is a generative model which is general and applicable to any kind of co-occurrence data, although here we limit the application to information retrieval for simplification. In the following, the document and term sets are denoted as $D = \{d_1, ..., d_N\}$ and $W = \{w_1, ..., w_M\}$, respectively, where N and M are the numbers of documents and terms in a training set. Documents are represented as sets of terms which they co-occur with. This representation is often called *bag-of-words* model.

The aspect model assumes the correspondence between the observed document-term co-occurrence $S = \{(d_1, w_1), ..., (d_L, w_L)\}$ and the unobserved latent classes $Z = \{z_1, ..., z_K\}$, where L is the length of the observed co-occurrence sequence and K is the number of the latent classes. In general, the number K may be smaller than the

number of documents or terms so that this model produces the compression effect compared to the original data.

The aspect model is built on the assumption that given a certain latent class, the occurrence of documents and terms is independent, that is:

$$P(d, w|z) = P(d|z)P(w|z). \tag{4.4}$$

The joint distribution over $D \times W$ is then given by

$$P(d, w) = \sum_{z \in Z} P(z)P(d|z)P(w|z), \tag{4.5}$$

which can be equivalently expressed as

$$P(d, w) = P(d) \sum_{z \in Z} P(z|d)P(w|z). \tag{4.6}$$

The generative process the equation (4.6) means can be explained as follows.

1. Choose a document d with the probability $P(d)$.
2. The degree of association of a latent class z to the document d is $P(z|d)$.
3. A term w can be considered as an expression of a latent classes concept. The degree of association between a term w and a latent class z is given by $P(w|z)$.

The latter (4.5) is a symmetric parameterization with respect to documents and terms, which is employed in the followings because of its simple implementation. Both probabilistic models are illustrated in Figure 4.2.

Note that parameters $P(z), P(d|z)$, and $P(w|z)$ (or $P(d), P(z|d)$, and $P(w|z)$ in the asymmetric model) are not directly observable from the corpus, thus we have to infer the parameter values. This should be done on a maximum-likelihood basis, but the values cannot directly computed from the observed data. In such a case, EM (expectation maximization) algorithm [Kita and Tsujii, 1999], which computes unobservable parameters in an iterative way, is employed. The following section gives the detail.

4.3.2 EM algorithm

Let the observable data be $\chi_{obs} = \{\vec{x}_1^{obs}, ..., \vec{x}_N^{obs}\}$, and the unobservable data $\chi_{mis} = \{\vec{x}_1^{mis}, ..., \vec{x}_N^{mis}\}$. N denotes the number of observations in this section. The aim of the EM algorithm is the maximum-likelihood estimation of the certain set of unknown parameters Θ. The target function to be maximized is

$$\begin{aligned} L(\Theta; \chi_{obs}) &= \log P(\chi_{obs}; \Theta) && (4.7) \\ &= \log \prod_i P(\vec{x}_i^{obs}; \Theta) && (4.8) \\ &= \sum_i \log P(\vec{x}_i^{obs}; \Theta), && (4.9) \end{aligned}$$

which is called *incomplete data log likelihood function.* "Incomplete data" stands for observable data — in this case, it is χ_{obs}.

The difference of $\log P(\vec{x}_i^{obs};\Theta)$ in (4.9) when the parameter $\Theta^{(t)}$ is updated to Θ on time t during the iteration is calculated as follows:

$$\begin{aligned}
& \log P(\vec{x}_i^{obs};\Theta) - \log P(\vec{x}_i^{obs};\Theta^{(t)}) && (4.10)\\
= \; & \log \frac{P(\vec{x}_i^{obs};\Theta)}{P(\vec{x}_i^{obs};\Theta^{(t)})} && (4.11)\\
= \; & \sum_{\chi_{mis}} P(\chi_{mis}|\vec{x}_i^{obs};\Theta^{(t)}) \log \frac{P(\vec{x}_i^{obs};\Theta)}{P(\vec{x}_i^{obs};\Theta^{(t)})} && (4.12)\\
= \; & \sum_{\chi_{mis}} P(\chi_{mis}|\vec{x}_i^{obs};\Theta^{(t)}) \times \log \left[\frac{P(\vec{x}_i^{obs},\chi_{mis};\Theta)}{P(\vec{x}_i^{obs},\chi_{mis};\Theta^{(t)})} \frac{P(\chi_{mis}|\vec{x}_i^{obs};\Theta^{(t)})}{P(\chi_{mis}|\vec{x}_i^{obs};\Theta)} \right] && (4.13)\\
= \; & \sum_{\chi_{mis}} P(\chi_{mis}|\vec{x}_i^{obs};\Theta^{(t)}) \log \frac{P(\vec{x}_i^{obs},\chi_{mis};\Theta)}{P(\vec{x}_i^{obs},\chi_{mis};\Theta^{(t)})} \\
& \qquad + \sum_{\chi_{mis}} P(\chi_{mis}|\vec{x}_i^{obs};\Theta^{(t)}) \log \frac{P(\chi_{mis}|\vec{x}_i^{obs};\Theta^{(t)})}{P(\chi_{mis}|\vec{x}_i^{obs};\Theta)}, && (4.14)
\end{aligned}$$

Here we define Q_i as:

$$Q_i(\Theta|\Theta^{(t)}) = \sum_{\chi_{mis}} P(\chi_{mis}|\vec{x}_i^{obs};\Theta^{(t)}) \log P(\vec{x}_i^{obs},\chi_{mis};\Theta). \tag{4.15}$$

Using Jensen's inequity below,

$$\begin{aligned}
\sum_x P(x) \log \frac{Q(x)}{P(x)} &\leq \sum_x P(x) \left(\frac{Q(x)}{P(x)} - 1 \right) \qquad (\because \log x \leq x - 1) \\
&= \sum_x Q(x) - \sum_x P(x) \\
&= 0, && (4.16)
\end{aligned}$$

the following inequity stands:

$$\sum_{\chi_{mis}} P(\chi_{mis}|\vec{x}_i^{obs};\Theta^{(t)}) \log \frac{P(\chi_{mis}|\vec{x}_i^{obs};\Theta^{(t)})}{P(\chi_{mis}|\vec{x}_i^{obs};\Theta)} \geq 0, \tag{4.17}$$

which produces

$$\log P(\vec{x}_i^{obs};\Theta) - \log P(\vec{x}_i^{obs};\Theta^{(t)}) \geq Q_i(\Theta|\Theta^{(t)}) - Q_i(\Theta^{(t)},\Theta^{(t)}). \tag{4.18}$$

The problem is, therefore, to find the value of Θ which maximizes $Q_i(\Theta|\Theta^{(t)})$. Because the preceding derivation is the case where only a single observation data $\vec{x}_i^{obs}$ is considered, we have to give

$$Q(\Theta|\Theta^{(t)}) = \sum_{\chi_{mis}} P(\chi_{mis}|\chi_{obs};\Theta^{(t)}) \log P(\chi_{obs},\chi_{mis};\Theta) \tag{4.19}$$

which considers all the observed data.

This function Q can be viewed as the calculation of the expectation of

$$L_C(\Theta;\chi) = \log P(\chi_{obs}, \chi_{mis};\Theta) \tag{4.20}$$

under the condition of $\chi_{obs}, \Theta^{(t)}$. As opposed to L in (4.9), L_C is called *complete log-likelihood function*, where "complete data" means all the data in the system, that is, $\chi = \chi_{obs} \cup \chi_{mis}$.

The steps of the EM algorithm is summarized as follows:

1. Give an initial value of $\Theta^{(0)}$.
2. Repeat the followings until Θ converges.
   (a) **E-step:** Calculate $Q(\Theta|\Theta^{(t)})$ of (4.19).
   (b) **M-step:** $\Theta^{(t+1)} = \arg\max_\Theta Q(\Theta|\Theta^{(t)})$

$P(\chi_{mis}|\chi_{obs};\Theta^{(t)})$ in (4.19) is derived using Bayes' theorem:

$$P(\chi_{mis}|\chi_{obs};\Theta^{(t)}) = \frac{P(\chi_{obs},\chi_{mis};\Theta^{(t)})}{\sum_{\chi_{mis}} P(\chi_{obs},\chi_{mis};\Theta^{(t)})}. \tag{4.21}$$

4.3.3 PLSI implementation

In the PLSI model, $P(z), P(d|z)$, and $P(w|z)$ in (4.5) are estimated using EM algorithm explained above. In other words, what corresponds to the unknown parameter Θ mentioned above is $P(z), P(d|z)$, and $P(w|z)(\forall z \in Z, \forall d \in D, \forall w \in W)$. The function Q in (4.19) is

$$\begin{aligned} Q(\Theta|\Theta^{(t)}) &= \sum_{i=1}^{L}\sum_{j=1}^{K} P(z_j|d_i,w_i;\Theta^{(t)}) \log P(d_i,w_i,z_j;\Theta) \\ &= \sum_{d\in D, w\in W}\sum_{j=1}^{K} n(d,w) P(z_j|d,w;\Theta^{(t)}) \log P(d,w,z_j;\Theta), \end{aligned} \tag{4.22}$$

where $n(d,w)$ is the frequency the term w appears in the document d. Here $P(z), P(d|z)$, and $P(w|z)$ which maximize Q are calculated by introducing Lagrange multipliers. From this procedure we obtain:

$$P(w|z) = \frac{1}{\alpha}\sum_{d\in D} n(d,w) P(z|d,w) \tag{4.23}$$

$$P(d|z) = \frac{1}{\beta}\sum_{w\in W} n(d,w) P(z|d,w) \tag{4.24}$$

$$P(z) = \frac{1}{\gamma}\sum_{d\in D}\sum_{w\in W} n(d,w) P(z|d,w), \tag{4.25}$$

where α, β, and γ are the coefficients for normalization which ensure $\sum_w P(w|z) = 1, \sum_d P(d|z) = 1$, and $\sum_z P(z) = 1$, respectively.

This technique is called "soft" clustering because the attribution probability distribution to all the classes is calculated for each term, whereas in "hard" clustering, each term is classified into only one class.

4.3.4 Tempered EM algorithm

Because the initial values of EM algorithm are chosen randomly and the update is done through iterations, there is no guarantee that the converged parameters are "globally" optimal values. Also, over-fitting, which aggravates the performance of the resultant language model, occasionally occurs when using EM algorithm to implement PLSI. To circumvent these drawbacks, Hofmann [Hofmann, 2001] proposes using *annealing* technique, which is closely related to deterministic annealing EM (DAEM) algorithm [Ueda and Nakano, 2002].

The essence of the DAEM algorithm is that the posteriori probability defined in (4.21) is not reliable in the early steps of iterations, thus it is replaced by a posteriori distribution $f(\chi_{mis}|\chi_{obs})$ by introducing inverse temperature β as:

$$f(\chi_{mis}|\chi_{obs};\Theta) = \frac{P(\chi_{obs},\chi_{mis};\Theta)^\beta}{\sum_{\chi_{mis}} P(\chi_{obs},\chi_{mis};\Theta)^\beta}. \tag{4.26}$$

The parameter β serves to anneal, or smoothen, the target function, so that the function has only a single global maximum when β is lower, which means the temperature is "high." Indeed, the distribution $f(\chi_{mis}|\chi_{obs};\Theta)$ equals to the uniform distribution when $\beta = 0$. Although original the DAEM algorithm "cools off" the annealing through iterations, gradually increasing the parameter β, we employed the *tempered EM algorithm* proposed by Hofmann [Hofmann, 2001], or TEM algorithm, where β is fixed through all the EM iterations. This is because of the complexity and the lack of practical stability of the DAEM algorithm, which we have found through preliminary experiments, too. Using the TEM algorithm, the final updating equations are:

$$P(w|z) = \frac{\sum_d n(d,w)\tilde{P}(z|d,w)}{\sum_{d,w} n(d,w)\tilde{P}(z|d,w)} \tag{4.27}$$

$$P(d|z) = \frac{\sum_w n(d,w)\tilde{P}(z|d,w)}{\sum_{d,w} n(d,w)\tilde{P}(z|d,w)} \tag{4.28}$$

$$P(z) = \frac{\sum_{d,w} n(d,w)\tilde{P}(z|d,w)}{\sum_{d,w} n(d,w)}, \tag{4.29}$$

where

$$\tilde{P}(z|d,w) = \frac{\{P(z)P(d|z)P(w|z)\}^\beta}{\sum_{z'}\{P(z')P(d|z')P(w|z')\}^\beta}. \tag{4.30}$$

The setting of the parameter β has a critical impact on the final performance of PLSI-based similarity calculation. We investigated the optimal value of β through preliminary experiment.

After running the (tempered) EM algorithm, we obtain the latent class distribution $P(z|w)$ using Bayes' rule. This can be considered as a distribution of latent semantics of a word type w. Thus, we can compute the semantic similarity/distance between word types w_1 and w_2 by calculating the distance between their corresponding distributions:

$$\mathrm{dist}_{\mathrm{plsi}}(w_1, w_2) = \mathrm{dist}(P(z|w_1), P(z|w_2)). \tag{4.31}$$

Again, there are several ways to calculate this distance between distributions. We compared a few of the previously introduced similarity/distance measures.

4.4 Experiments

In this section, we conduct experiments to compare all the models explained in this chapter, i.e., LSI and PLSI, with the distributional similarity model.

A large part of the experimental conditions is the same as the previous one — we used the same corpus and context extraction method. However, the models in this chapter, especially LSI, are computationally intensive, thus it is impossible to deal with such large co-occurrence data, which have 15,110 word types and 34,570 context types, as we dealt with in the previous chapter. Therefore, we applied the context feature selection technique beforehand to reduce the dimensionality of the original input data. Specifically, we used the DF-based context type selection, the detail of which is described in Section 6.4, and the dimensionality is reduced to 20% of the original. This selection scheme detects which context types have the lowest DF values and delete them, assuming that low-DF context types have low contribution to the overall performance. This method is to be shown effective in reducing the number of context types while keeping the performance loss at minimum, as demonstrated in Chapter 6.

As for the similarity measure and weighting function, we used the three best performing measures in the previous experiments, namely `cosine`, `jaccard-v`, and `js` for the former, and `pmi` for the latter.

4.4.1 Effect of Inverse Temperature

Before using PLSI as a similarity model, we need to see how the inverse temperature parameter β of the TEM algorithm affects the overall performance of PLSI, and need to fix it to the optimal value based on a preliminary experiment. In this experiment, we vary β from 0.7 to 1.0 with an interval of 0.02, and investigate how the performance (AP and CC) changes accordingly. The number of the latent classes K is fixed to $K = 200$ in this experiment.

Figure 4.3 shows the result. Because the effect of β might be dependent on the similarity measures, we used all the three measures (`cosine`, `jaccard-v`, and

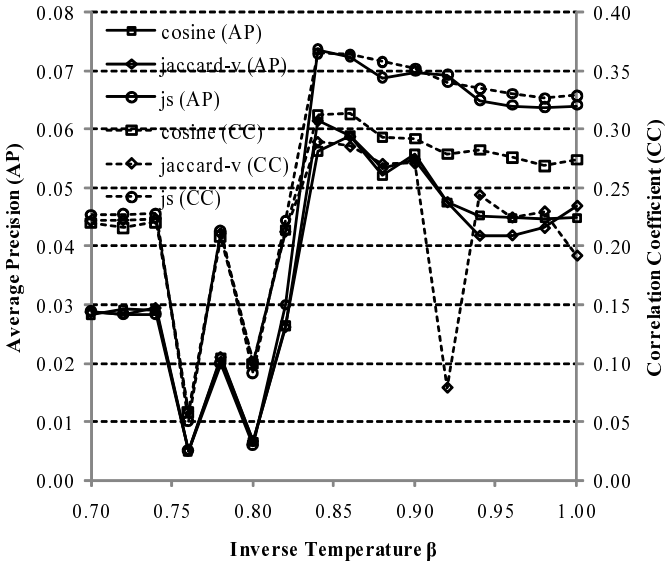

Figure 4.3: Change of performance v.s. the inverse temperature β

`js`) for comparison. It is clearly shown that the impact of β is significant and the performance shows a sudden increase and a peak around $\beta = 0.84$. Similar behavior of β is also reported in [Hofmann, 2001]. Also, the result turned out to be independent of similarity measures. We decided that the peaks of AP and CC, though slightly different in positions, are close enough to each other to fix the parameter to $\beta = 0.84$ to be used in the following experiments.

4.4.2 Performance Comparison

In this section we compare all the three models of distributional similarity — the naïve, vector space model (VSM; Chapter 2), LSI (Section 4.2), and PLSI (Section 4.3). The dimensionality K is varied from 50 to 1000 in 50 increments for LSI and PLSI, while the maximum AP and CC values are recorded. The result is shown in Table 4.1.

In Table 4.1, the maximum numbers of AP and CC among the different models with the same similarity measure are emphasized in bold face. It suggests that AP is greatly dependent on the similarity measure in use — `cosine` and `jaccard-v` perform well for VSM and LSI, while `js` is the best for `PLSI`. This might be because `cosine` and `jaccard-v` were designed for vector-based models in nature, while `js` was for probability-based models. These measures perform best with their suitable models, as this experiment shows.

On the other hand, the result for CC is quite straightforward — it clearly shows the superiority of PLSI, regardless of the similarity measure in use. This might be related to the fact that CC is more susceptible to "non-related pairs," which is fully discussed in Chapter 6.

Table 4.1: Comparison of distributional similarity models

Model	Similarity Measure	AP	CC
VSM	`cosine`	**0.1193**	0.3316
	`jaccard-v`	0.1198	0.3126
	`js`	0.0378	0.3306
LSI	`cosine`	0.1161	0.1842
	`jaccard-v`	**0.1260**	0.1917
	`js`	0.0719	0.0814
PLSI	`cosine`	0.0641	**0.3362**
	`jaccard-v`	0.0625	**0.3192**
	`js`	**0.0759**	**0.3711**

To sum up, it is difficult to definitively say which model is the best for the distributional similarity task. The simplest, VSM may work well as long as the appropriate similarity measure is chosen, while PLSI may also work better, especially for CC, when the probability-based similarity measure is used.

4.4.3 Effect of Dimensionality Reduction

Because we are now interested in how the number of the latent classes K affects the performance of LSI and PLSI, we conducted an experiment, where we compared the performance change of LSI and PLSI with respect to K. We chose LSI+`jaccard-v` and PLSI+`js` as the representative cases where these latent semantic models perform the best.

Figure 4.4 shows the result when we changed K from 50 to 1000 with a step of 50. It is evident that PLSI and LSI distinctly behave with respect to K — PLSI shows clear peaks at a small number of K, while LSI only shows a clear peak of CC at $K = 100$. We also notice that the peak of CC is shifted toward lower value of K compared to AP. This trend is also observed in the context selection task in Chapter 6, where the detailed cause is discussed. This result suggests that the performance peaks of LSI and PLSI are, if any, achieved at relatively lower value of K, around 100 to 150.

4.5 Conclusion

In this chapter, we applied these latent semantic models, i.e., LSI and PLSI, to the synonym acquisition problem, and compared their performances. After we introduced these models, we firstly investigated the effect of the inverse temperature parameter β, and showed that it showed a clear peak around $\beta = 0.84$. We secondly compared the best performances of VSM, LSI, and PLSI, and found out that, although each model behaved quite differently, every model had its own suitable similarity measure, and LSI+`jaccard-v` performed well for AP and PLSI+`js` did for CC. Finally

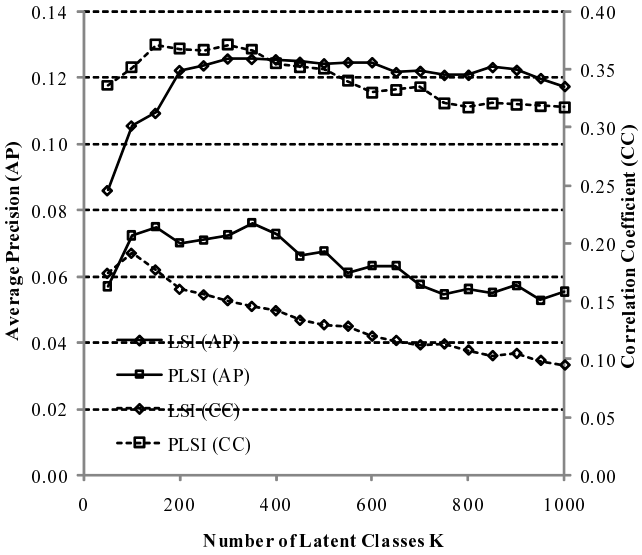

Figure 4.4: Performance of LSI and PLSI v.s. the number of the latent classes K

we investigated the effect of the dimensionality reduction. The experimental result suggested that the performance showed peaks around $K = 100$ to 150, although they were dependent on the models and evaluation measures (AP and CC).

As we mentioned in Section 1.2, there are a number of related models of document-term/word-context co-occurrence, such as Latent Dirichlet Allocation (LDA) [Blei et al., 2003], Random projection [Bingham and Mannila, 2001], and Semantic kernels [Kandola et al., 2002]. The application of those models to the synonym acquisition task is the future work.

Chapter 5

Context Extension

5.1 Introduction

The first step of distributional similarity calculation is *context extraction*, where a corpus is analyzed and contexts are extracted. Although we have been using the dependency structure as contexts so far in this thesis, there is plenty of freedom in choosing which contextual information to use. For example, as we showed in Section 2.2, we could also use context types such as "the next word is *of*," "located at the 6th position in a sentence," or even "the sentence starts with a letter *T*." We are sure that most of these context types are undoubtedly unuseful for word characterization. However, the importance of the choice of contextual information to adopt has been considerably underestimated in the previous work. Almost no attention has been paid to what kind of contextual information, or their combinations, is useful for distributional similarity. However, as [Curran and Moens, 2002b, Hagiwara et al., 2006] have pointed out, the choice of useful contextual information is considered to have a critical impact on the performance of any methods utilizing distributional similarity. Therefore, further investigations on which types of contexts are essentially contributing are strongly required.

So far, *word-based context* and *dependency* are the most widely adopted contextual categories. Many studies have shown that dependency relation is more effective for lexical knowledge acquisition compared to word-based context. However, dependency relations are usually limited to two words sharing direct dependency between them, thus it sometimes misses important word relations. In response to this problem, we propose the use of *indirect dependency* — the extension of normal *direct* dependency — as one of the general ways to enhance contextual information for distributional similarity to improve performance. This study investigates its effect to the performance and various parameter settings when applying it to actual tasks. More specifically, we firstly formalize the context which incorporates indirect dependency, composed from two or more contiguous dependency relations, and show its effectiveness to the performance when compared to the conventional *direct dependency* for the synonym acquisition task. The performance evaluation of synonym acquisition is based on average precision (AP) and correlation coefficient (CC). We then pay attention to the

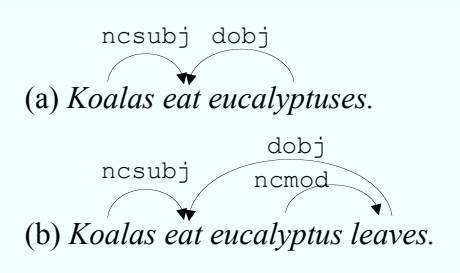

Figure 5.1: Examples of indirectly related words

context representation of indirect dependency, which has been underestimated in the previous studies, although it is one of the important parameters when constructing semantic spaces. We then compare the individual performance.

This chapter is organized as follows: In Section 5.2, we mention the background of how we come up with the idea of indirect dependency, and formalize it. Section 5.3 provides the experimental conditions and results, with the first part regarding the effectiveness of indirect dependency, and the second part being about the four kinds of context representation for indirect dependency: terminal word `TW`, word path `WP`, syntactic path `SP`, and full path `FP`. Section 5.4 concludes this chapter.

5.2 Indirect Dependency

In this section, we firstly describe the limitation of normal direct dependency and how we came up with the idea that the extension of direct dependency can be beneficial. Next, the formalization of indirect dependency, which is used in this chapter, is described.

5.2.1 Limitation of Direct Dependency

Although previous studies have shown that syntax-based context is capable of choosing highly effective contexts, we notice that the coverage of syntax-based context is rather limited, which can be problematic when fully utilizing the contexts. To confirm this, we conducted a simple preliminary experiment, where we compared the performances of word-based context `wbc` and syntactic-based context (we call this as `dep` in this chapter, since we used dependency structure as syntax-based context). The result showed, while `dep` certainly outperformed `wbc`, there is only a slight performance difference between them. We suspect that this is because `dep` only covers two words with direct dependency. It can miss a large portion of important word relations, causing the performance loss.

We show this using very simple examples illustrated in Fig. 5.1. It is obvious that there is a strong semantic relation between the words *koala* and *eucalyptus*, although this relation cannot be obtained using `dep` in neither the sentence (a) nor (b), while `wbc` easily represents it as the relation *koala - eucalyptus*.

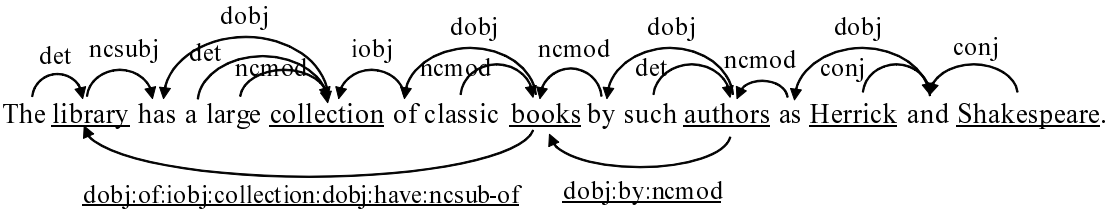

Figure 5.2: Example of Direct and Indirect Dependency

The problem here is that these two words, *koala* and *eucalyptus*, are related only *indirectly* via two (in the sentence (a)) or three (in the sentence (b)) dependency relations. Such kinds of indirect relations are included in `wbc` with the window radius of 3 but not in `dep`. We suspect that this kind of difference caused the performance loss for `dep`. To overcome this problem, we came up with the idea to include these *indirect dependency* relations, i.e., words that can be reached by following two or more dependency relations, and enhance the contextual information for distributional similarity to improve the performance. The detailed method for this extension is described in the following section.

5.2.2 Formalization

As dependency is often represented as relations, it is useful and straightforward to formalize indirect dependency as the composition of relations. Here we consider the dependency relations in a certain sentence s as a binary relation D over $W = \{w_1, ..., w_n\}$ i.e., $D \subseteq W \times W$, where $w_1, ..., w_n$ are the word tokens in the sentence s. Since no words can be dependent or modifier of itself, D is irreflexive.

Dependency relations generally have labels such as `subj` and `dobj` which specify what kind of syntactic relations head and modifier possess. We encode this information as equivalence classes of D, which means that D is partitioned into the equivalence classes D_{r_i} such that $D = \bigcup_{i=1}^{k} D_{r_i}, D_{r_i} \cap D_{r_j} = \phi$ if $i \neq j$, based on the labels $r_1, ..., r_k$. Note that both the head-modifier (e.g., `ncsubj`) and the corresponding modifier-head (e.g., `ncsubj-of`) relationships are included in D and the direction information of dependency is incorporated into the labels.

As the next step, we define the composition of dependency $D^2 = D \circ D$ as *indirect dependency* where two words are related via two dependency relation edges. When an indirect dependency relation $d = (w_i, w_j) \in D^2$ is composed from an edge $(w_i, w_m) \in D_{r_i}$ and another one $(w_m, w_j) \in D_{r_j}$, the label of d, i.e., which equivalence class d belongs to, is also composed as $r_i : w_m : r_j$. In other words, the labels are the concatenations of all the labels and words on the path from w_i to w_j. We also define multiple composition of dependency recursively: $D^1 = D, \forall l > 1.\ D^l = D^{l-1} \circ D$. These are also indirect dependency relations in a broad sense. Notice here that D^l $(l > 1)$ can generally include reflexive relations, although it is clear that such relations do not serve as useful word features. Thus, we re-define the composition operation so that the composed relation does not include any reflexive edges, i.e.,

$D \circ D - \{(w, w) | w \in W\}$. The contexts for indirect dependency $D^1, D^2, ...$ are referred as `dep1`, `dep2`, and so on.

Some examples of indirect dependency are shown below the sentence in Fig. 5.2. By composing $(authors, by) \in D_{\text{dobj}}$ and $(by, books) \in D_{\text{ncmod}}$, we obtain an indirect dependency $(authors, books) \in D_{\text{dobj:by:ncmod}} \subseteq D^2$, which we consider as a semantically important lexical relation in the sentence. Similarly, the composition of four contiguous dependency edges between word *books* and *library* gives $(books, library) \in D_{\text{dob:of:iobj:collection:dobj:have:ncsubj}-\text{of}} \subseteq D^4$, which is also an important lexical semantic relation in the sentence.

5.3 Experiment

Now we describe the evaluation results for indirect dependency, and the detailed comparison of the acquisition and context parameters in this section.

5.3.1 Condition

The experimental settings are almost the same as the ones we used in Chapter 3. We used the 1994 New York Times articles, which were then parsed by RASP2 and converted to the dependency relation D as we described in Section 2.2. To reduce the computational complexity, we adopted $\theta_w = \theta_c = 100$.

As the basis of the semantic space, the second component of every indirect relation, i.e., w_j in $(w_i, w_j) \in D^n$, was used. For example, `books` was used as the context for the word *authors* in $(authors, books) \in D_{\text{dobj:by:ncmod}} \subseteq D^2$.

As listed in Sections 2.3 and 2.4, there are many weight functions and similarity measures to handle, let alone their combinations. Because the purpose of this chapter is the investigation of the indirect dependency effectiveness, we limit the target, instead of adopting all these combinations, and selected only a few practical combinations of weights and measures, that is: `tfidf+cosine`, `pmi+jaccard-s`, and `ttest+jaccard-v|`. We chose `tfidf+cosine` because it is the most widely adopted combination of weight and measure function for information retrieval. The second one, `pmi+jaccard-s` is also common for similarity-based task [Weeds et al., 2004], and individual effectiveness of PMI and Jaccard coefficient has been shown [Curran and Moens, 2002a]. The weight `ttest` and the metric `jaccard-v` are also shown to be one of the best performing in [Curran and Moens, 2002a], thus the last combination `ttest+jaccard-v` was chosen for the diversity. We call these three combinations *calculation parameters* in the following.

5.3.2 Effect of Indirect Dependency

Now we compare the performances of direct and indirect dependency contexts. Figure 5.3 shows the performances of synonym acquisition when the calculation parameter is `tfidf+cosine`, `pmi+jaccard-s`, and `ttest+jaccard-v`. They compare the five `dep`

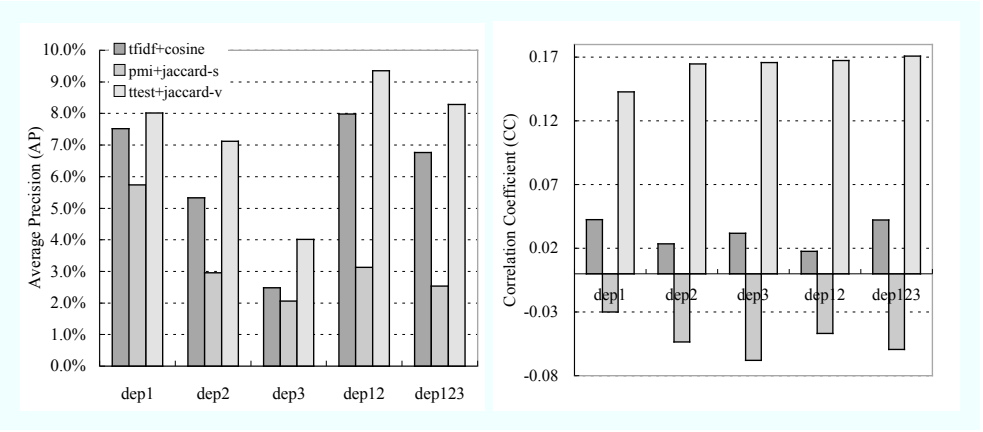

Figure 5.3: Comparison of Direct and Indirect Dependency Performance

categories and combinations: C^1(`dep1`), C^2(`dep2`), C^3(`dep3`), $C^1 \cup C^2$(`dep12`), and $C^1 \cup C^2 \cup C^3$(`dep123`), where C^n is the set of context types extracted from D^n.

The overall observation is that adding the indirect dependency `dep2` to the conventional direct dependency `dep1`, i.e., `dep12`, increased the performance in some cases, and in most cases `dep2` did not outperform `dep1` by itself. More specifically, this tendency is dependent on the calculation parameters — for `ttest+jaccard-v` the performance increased whenever `dep2` was added to `dep1`. For `tfidf+cosine`, `dep2` had negative effects on CC values, although the absolute value of performance was low (around 0.04) in the first place. This effect can be ignored.

To look further into the effect of indirect dependency, we list some actual examples of the acquired synonyms in Table 5.1, in the descending order of similarity. The table also compares the cases when `dep1` and `dep12` were used as context. The shown result is the case when the acquisition was conducted using `ttest+jaccard-v` as calculation parameter. The overall results of the top-ranked relevant words looked quite encouraging, showing a small number of totally irrelevant words in the list, which are marked by asterisk marks based on subjective judgement. Furthermore, some of the irrelevant words appearing in the `dep1` lists disappeared and were replaced by more relevant words to the target words in the `dep12` list, as *mussel* and *nudity* for the target word *language*, as well as *centimeter* and *convenience* for the target word *jewelry* show. We can grasp the general trend that `dep12` improves the quality of acquired synonyms from the list.

Some of the comparisons, including the one mentioned above, shows partial effectiveness of indirect dependency, although the overall result does not look encouraging. More specifically, 3 out of 6 (two evaluation measures and three calculation parameters) comparisons showed the negative effect of `dep12` over `dep1`. The result is considerably dependent on the choice of calculation parameters.

Now remember that we used the *terminal word* positioned at the end of each dependency path as the contexts, i.e., bases of the constructed semantic space. However, we suspect that this kind of context representation is too coarse to fully represent

Table 5.1: Examples of Acquired Synonyms for dep1 and dep12

target word = *language*				target word = *jewelry*			
`dep1`		`dep12`		`dep1`		`dep12`	
word	sim.	word	sim.	word	sim.	word	sim.
tongue	0.1322	dialect	0.1798	necklace	0.2478	furniture	0.1528
translation	0.0964	subtitle	0.1694	centimeter*	0.1435	accessory	0.1238
word	0.0947	literature	0.1494	medalist	0.1327	diamond	0.1231
religion*	0.0923	translation	0.1419	medal	0.1191	wearing	0.1227
literature	0.0882	translator	0.1063	earring	0.1162	clothing	0.1107
subtitle	0.0833	tongue	0.1020	clothing	0.1143	earring	0.1098
mob*	0.0810	culture*	0.0961	artifact	0.1186	textile	0.1004
text	0.0794	word	0.0949	watch	0.1112	clothes	0.0924
volume	0.0773	religion*	0.0913	lighting*	0.0999	apparel	0.0906
villager*	0.0709	pop*	0.0804	meter*	0.0991	cologne	0.0900
truth	0.0601	interpreter	0.0771	bracelet	0.0961	necklace	0.0880
explorer*	0.0598	explorer*	0.0758	convenience*	0.0953	shoe	0.0871
mussel*	0.0587	text	0.0718	pin	0.0946	silver	0.0863
lesson	0.0575	lesson	0.0711	hardware	0.0941	retailer*	0.0813
proficiency	0.0567	accent	0.0699	heist*	0.0928	pendant	0.0812
wrong*	0.0552	situation*	0.0689	sneaker	0.0906	copper	0.0806
nudity*	0.0547	history*	0.0678	mining	0.0845	watch	0.0782
who*	0.0525	profanity	0.0672	furniture	0.0815	bronze	0.0780
method*	0.0519	proficiency	0.0658	shoe	0.0795	heist*	0.0758
literacy	0.0515	initial*	0.0616	underwear	0.0778	furnishings	0.0749

the detailed difference of syntactical relations of words. Get back to the example illustrated in Fig. 5.1, and take the sentence (a) for example. When the relation: $(koala, eucalyptus) \in D_{\text{ncsubj:eat:dobj}-\text{of}} \subseteq D^2$ in this sentence is used, the context for *koala* will be `eucalyptus`. However, it can be important that the relation is not "Koalas *grow* eucalyptuses," nor "Koalas *hate* eucalyptuses," but "Koalas *eat* eucalyptuses." This kind of word relation cannot be encoded in this *terminal word* representation. The mediating word *eat* should also be included in this case. Furthermore, there is a big discrepancy in the meaning between the sentences "Koalas eat eucalyptuses" and "Eucalyptuses eat koalas." These relations are degenerated into an identical relation `koalas - eat - eucalyptus` even if the mediating word *eat* is included, unless syntactic relations of words are taken into consideration. Thus we suppose that much richer context representations which incorporate mediating words and/or syntactic relations have positive effects on the acquisition performance, and will discuss this issue in the following sections.

Table 5.2: Context Representations for Direct and Indirect Dependency
Examples of context representations are for the word *authors* in
$(authors, books) \in D_{\text{dobj:by:ncmod}} \subseteq D^2$.

Representation	Expression	Example (for *authors*)
Terminal Word (`TW`)	w_l	`books`
Word Path (`WP`)	$w_1 : w_2 : ... : w_{l-1} : w_l$	`by:books`
Syntactic Path (`SP`)	$r_1 : r_2 : ... : r_l : w_l$	`dobj:ncmod:books`
Full Path (`FP`)	$r_1 : w_1 : r_2 : w_2 : ... : w_{l-1} : r_l : w_l$	`dobj:by:ncmod:books`

5.3.3 Context Representation

In this section, we consider four kinds of context representations when converting indirect dependency formalized in Section 5.2.2 to contexts. This representation corresponds to the *basis mapping function* μ which maps dependency paths into dimensions in semantic space described in [Pado and Lapata, 2007], although the function and its effect have not been discussed so far.

The four context representations considered are summarized in Table 5.2. In the following, suppose an indirect dependency $(w_0, w_l) \in D_r \subseteq D^l$, where the label $r = r_1 : w_1 : r_2 : ... : w_{l-1} : r_l$. Here we define the sequence of words $w_0, w_1, ..., w_l$ as *dependency path* and l as the length of the path. We call w_0 as *target word*, whose contexts are to be obtained, and w_l, i.e., the word positioned at the end of the path, as *terminal word*.

Terminal Word (TW) This is the simplest context representation which has been used so far, where an indirect dependency is represented by the terminal word w_l. For example, from the indirect dependency example: $(authors, books) \in D_{\text{dobj:by:ncmod}} \subseteq D^2$, the context for the target word *authors* is `books` and vice versa, regardless of the syntactic relations these two words have. Because the context is represented only by the terminal word no matter how long the dependency path is, this is the coarsest context representation of the four, and is the same as the one adopted in [Pado and Lapata, 2007], where it is referred as *word-based basis mapping*.

Word Path (WP) In addition to the terminal word (`TW`), this representation, word path (`WP`), takes into account all the mediating words along the dependency path. The context for the target word w_0 is represented as $w_1 : w_2 : ... : w_{l-1} : w_l$ (`by:books` for the example shown above). Notice that this `WP` representation is one form of equivalence partitioning of `TW` based on the concatenation of mediating words.

Syntactic Path (SP) This representation takes into account the terminal word and the syntactic relations the two words have. Here the context for the target word w_0 is represented as $r_1 : r_2 : ... : r_l : w_l$. For example, the context for the target word *authors* is represented as: `dobj:ncmod:books`. In turn, the context for *books* is `ncmod-of:dobj-of:authors` and the direction of the path is completely opposite.

Table 5.3: Statistics of the Constructed Semantic Spaces

Representation	Context	Words	Contexts	Co-occurrences
	wbc	11,638	25,671	6,625,061
TW	dep1	7,671	7,093	1,705,931
	dep2	8,524	9,667	3,314,953
	dep3	9,454	11,574	4,266,411
WP	dep1	7,671	7,093	1,705,931
	dep2	8,519	23,470	3,439,624
	dep3	9,371	8,249	901,044
SP	dep1	7,688	11,051	1,916,329
	dep2	8,654	20,150	4,019,586
	dep3	9,603	24,578	4,698,221
FP	dep1	7,688	11,051	1,916,329
	dep2	8,646	19,792	2,527,141
	dep3	9,351	4,986	516,509

Notice that this SP representation is another form of equivalence partitioning of TW based on the concatenation of syntactic relations.

Full Path (FP) The finest representation of context is full path (FP), where the mediating words of indirect dependency are also considered in addition to the syntactic relations and the terminal word. More specifically, the context for the target word w_0 is represented as $r_1 : w_1 : r_2 : w_2 : ... : w_{l-1} : r_l : w_l$ (dobj:by:ncmod:books for the example shown above). Notice that this FP representation is the equivalence partitioning of both WP and SP.

5.3.4 Comparison of Context Representation

Extracted Semantic Spaces

Before moving on to the result for context representations, in Table 5.3 we show the statistics, i.e., the numbers of unique word and context types, and co-occurrences, of the constructed semantic spaces using three kinds of contextual categories, three steps of indirect dependency, and four forms of context representations. It is shown that incorporating indirect dependency greatly increases the variation of both contexts and co-occurrences (compare dep1-WP and dep2-WP, for instance). On the other hand, when the indirect dependency was extended to dep3, the numbers of context types and co-occurrences actually decreased for some of the context representations (WP and FP). We attribute this to the context representation granularity — these two representations are so fine-grained that most of the contexts were removed because they occurred less than θ_c times, causing the decreases in the semantic space sizes. As for the wbc, we observe the data size problem here too — the size of wbc contexts is greater than most of the context settings, although the performance of wbc does not exceed that of dep, as the previous experiment showed.

Context Representations

Now we compare the individual performance of context representations. Figs. 5.4, 5.5, and 5.6 show the comparisons of direct and indirect dependency when the calculation parameter is `tfidf+cosine`, `pmi+jaccard-s`, and `ttest+jaccard-v`, respectively. Notice that the representations `TW` and `WP`, as well as `SP` and `FP` for `dep1` are the same, thus the corresponding performances are also identical.

We can tell from the result that context representation is such an important factor that it greatly influences the performances and the effectiveness of indirect dependency. More specifically, for the two complex context representations, i.e., `WP` and `FP`, `dep2` gave positive effect and increased performance in as many as 10 out of 12 comparisons between `dep1` and `dep12`. However, other two simpler context representations, i.e., `TW` and `SP`, the performance increase was observed for 8 out of 12 comparisons. This result is summarized as this: the finer the representation of indirect dependency is, the better the result is likely to be. This tendency is especially salient for the AP values of `pmi+jaccard-s`, where the performances for `TW` and `SP` decreased while that of `WP` and `FP` increased by adding indirect dependency `dep2`.

This provides us with an important suggestion that, in order to make the most of indirect dependency, it is preferable to use more complex and richer forms of context representations such as `WP` and `FP` in constructing the basis for semantic spaces. Furthermore, the data size comparison showed that finer representations do not always increase the semantic space size (Table 5.3), thus there is no need to worry about the trade-offs between data size and performance.

Considering all these results, we observe that the order of well-performed representations was FP > WP > SP > TW, by comparing each group of four bars with the same parameters in the graphs. There are only a few exceptions to this, including AP values for `tfidf+cosine`, where simpler representations such as `TW` and `SP` often performed better than more complex ones. It is interesting that this order is exactly the same as the granularity scale, i.e., the complexity or the richness of representation. This result also confirms the superiority of fine-grained context representations in general.

5.3.5 Comparison of Other Parameters

Effect of Multiple Steps of Indirect Dependency

We confirmed the performance improvement for 18 out of 24 comparisons (2 performance measures, 3 parameter settings, and 4 context representations) of before and after the `dep2` addition, thus we can conclude that incorporating indirect dependency is effective on the whole for synonym acquisition tasks. However, extending the dependency to `dep3` worsened the performance for most cases, and adding `dep3` to `dep12`, i.e., using `dep123`, did not further improve the result, either. This result suggests that extending and augmenting direct dependency just one step is sufficient in practice.

Calculation Parameters

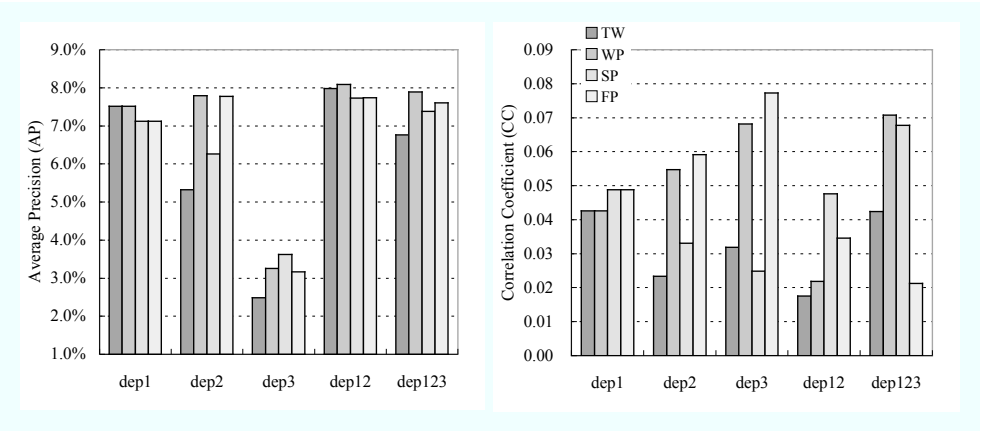

Figure 5.4: Comparison of Direct and Indirect Dependency for `tfidf+cosine`

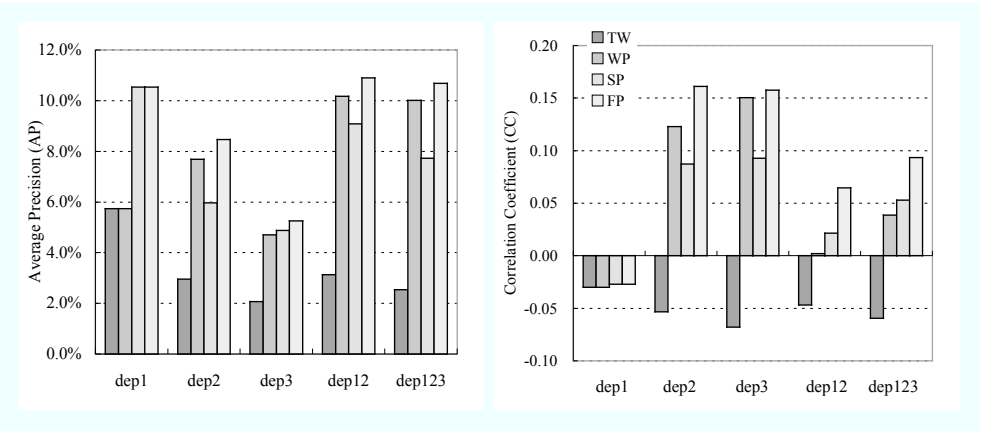

Figure 5.5: Comparison of Direct and Indirect Dependency for `pmi+jaccard-s`

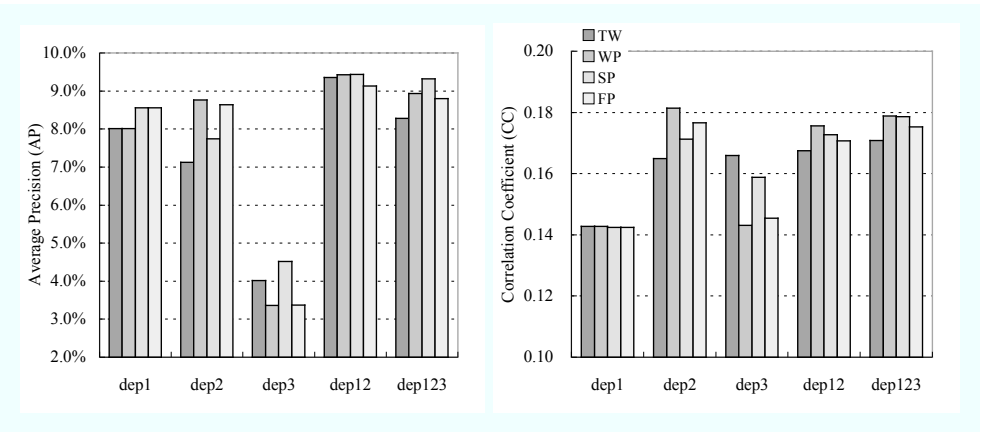

Figure 5.6: Comparison of Direct and Indirect Dependency for `ttest+jaccard-v`

Lastly, we briefly mention the effect of calculation parameter, which turned out to be another important factor when acquiring synonyms automatically. While `pmi+jaccard-s` performed the best for AP, CC values for `pmi+jaccard-s` were exceptionally "sensitive" to the performance differences. It is known that the weight function `pmi` tends to detect interesting and rare results [Curran and Moens, 2002a]. We suspect that this property affected negatively in this case. The parameter `ttest+jaccard-s`, on the contrary, showed stable and relatively good performance. On the whole, the comparison of Figs. 5.4, 5.5, and 5.6 suggests that `pmi+jaccard-s` and `ttest+jaccard-v` were the best choices. This result goes roughly along with Curran and Moens' comparative study [Curran and Moens, 2002a].

5.4 Conclusion

In this chapter, we extended the normal direct dependency to cover indirectly related words and enhance the contextual information for distributional similarity. Word dependency was defined as a binary relation over the sentence words, and this indirect dependency was formalized as the composition of dependency relations. We also proposed four kinds of context representations, which have been underestimated in the previous studies utilizing dependency-based contexts.

We investigated the effectiveness of indirect dependency using automatic synonym acquisition. The experiment showed that incorporating indirect dependency in addition to direct dependency was effective for the acquisition performance. The improvement was especially clear when fine-grained context representations, namely `WP` and `FP`, were used.

When compared to the conventional word-based context, even higher performance was achieved by dependency-based context with much smaller data size, which means that the indirect context is effective in terms of quality as well as computational cost. The use of indirect dependency, along with the fine-grained context representations, is a very efficient way to increase the contextual information variety, taking into consideration the fact that the diversity of the contexts is likely to be essential to the acquisition performance.

As introduced in Chapter 4, several methods which utilize latent analysis techniques such as LSI [Deerwester et al., 1990] to compress the original dimensionality of a semantic space were proposed and shown effective. The contextual extension proposed in this chapter does not conflict with these techniques. Rather, they can be incorporated and used at the same time, and may result in even better outcome, which should be investigated in the near future.

As the future work, the effectiveness of our methods and context representations should be confirmed using other kinds of applications and other word relations than synonyms. Also, because where we started from this time is the "difference" of word-based contexts and dependency-based contexts, the investigation of other kinds of useful contextual information and their comparison should be conducted as the future work.

In this chapter, we treated all the context types equally. However, we suppose that

there are effective contexts as well as useless ones, and different weights should be assigned to different context types. Pado and Lapata [Pado and Lapata, 2007] tackled this issue by introducing the *path value function*, which assigns different weights to the paths using parameters such as path length and word relations. Selecting important context types is an important issue itself, which is to be discussed in detail in the next chapter.

Chapter 6

Context Selection

6.1 Introduction

As we have seen in the previous chapters, most of the acquisition methods can be regarded as the combination of these two steps: (1) context extraction, and (2) similarity calculation. In the first step, the methods extract useful contextual information of words to characterize the target words from large corpora. In the second step, word similarity is calculated using the extracted contexts in the previous step.

A wide range of contextual information has been utilized for distributional similarity calculation, achieving considerable success. This includes word-based context, which uses surrounding words as context [Lowe and McDonald, 2000]
and [Curran and Moens, 2002b], dependency-based context, which uses words having syntactical relations with the target words as context [Hindle, 1990, Ruge, 1997, Lin, 1998a], and dependency path, which uses paths in a dependency tree as context [Lin and Pantel, 2001, Pado and Lapata, 2007]. However, a major problem which arises when adopting distributional similarity is that it easily yields a huge amount of context types and co-occurrences. This can lead to high dimensionality of semantic spaces, often up to the order of tens or hundreds of thousands, which makes the calculation computationally impractical. This problem poses a real challenge especially for machine learning techniques. A few researchers have begun to apply supervised learning methods such as naive bayes and logistic regression to lexical relation acquisition [Snow et al., 2004], although an excessively high dimensionality makes the computation almost impossible in many cases. However, not all of the contexts are necessarily useful, and some of them are even harmful and act as noise, worsening the calculation performance. Therefore, it is desirable to identify the contributing contexts and to remove the poorly-performing ones, before the similarity calculation is taken on. This allows to ease the expensive cost and to eliminate noise to improve performance.

However, whereas a few studies which tackled context or co-occurrence selection problem in a limited way can be found, there are no qualitative measures proposed or comprehensive study conducted for this problem. Therefore, general methods based on quantitative measures which can be used for reduction and selection of any kind

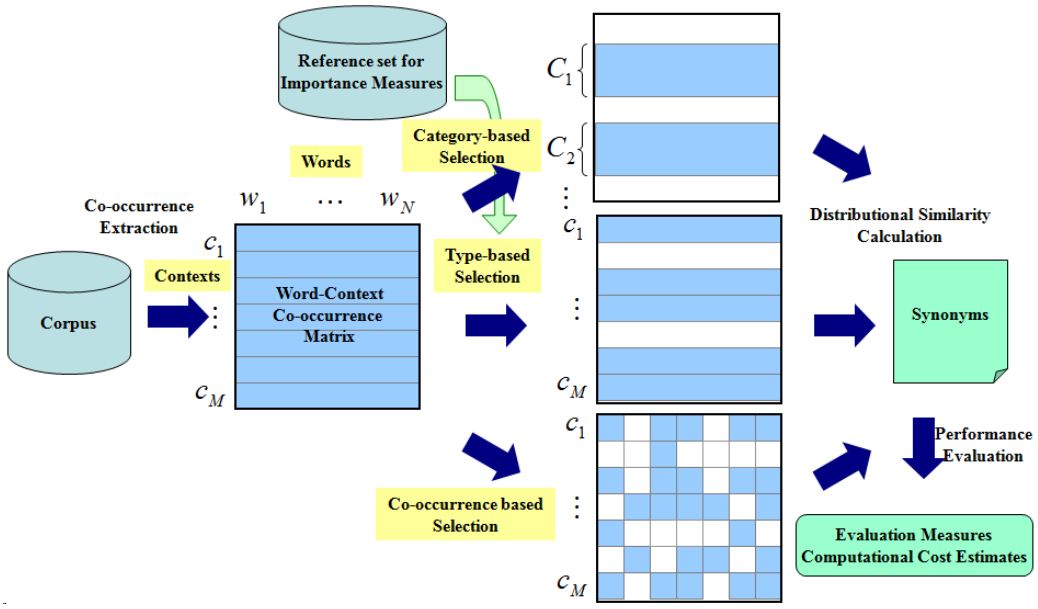

Figure 6.1: Overview of Selection Schemes

of context categories, types, and co-occurrences are strongly required. In response to this problem, in this chapter we propose three different schemes of context selection for distributional similarity, namely, category-based, type-based, and co-occurrence based selection, and show their effectiveness in selecting well-performing contexts, in terms of trade-off between performance and computational cost reduction.

The overview of three selection schemes is described in Fig. 6.1. The overall context selection process is as follows: firstly, the corpus is analyzed and all the co-occurrences of words and context types are extracted. Secondly, extracted context types and/or co-occurrences are selected based on one of the following three selection schemes: category-based, type-based, and co-occurrence based selection. Finally, distributional similarity is calculated using only the remaining co-occurrences and synonyms are acquired.

Category-based selection, the simplest and conventional selection scheme, limits the variety of context based on syntactic category. While there are various kinds of contextual information proposed so far, we especially paid attention to two classes: word-based context (`wbc`) and dependency-based context (`dbc`). For `wbc`, i.e., the context based on surrounding words, we set up categories according to relative positions of context words. For `dbc`, the adopted categories are based on grammatical relations (GRs) [Briscoe et al., 2002] of dependency relation.

The major drawbacks of category-based selection are the fact that categories must be fixed beforehand, and contributions of individual context types remain unknown. It is desirable that we can determine the importance of contexts on a type-by-type basis. Here, shifting our attention from synonym acquisition to other areas, a great deal of studies on feature selection has been conducted in the literature, especially for text categorization [Yang and Pedersen, 1997] and gene expression classification

[Ding and Peng, 2003]. Whereas these methods have been successful in reducing feature size while keeping classification performance, the model of distributional similarity is radically different from that of classification. Whether the same methods are applicable to and effective for automatic context type selection for word similarity problems is yet to be investigated. In this chapter we solve this problem by introducing a new formalization of the distributional similarity, where it is regarded as a classification problem based on word pairs, not individual words. This enables the importance calculation of context types, leading to the finer-grained, *type-based selection*.

The third and last selection scheme is *co-occurrence based selection*, the generalization of Curran's method based on canonical attributes [Curran and Moens, 2002a], which assigns an importance score to each co-occurrence of words and contexts. Although it is difficult to know the contribution of each context type compared to type-based selection, this enables the finest-grained selection of contexts.

We introduce importance measures which can be used for type- and co-occurrence based selections, then investigate how well they work for removing unwanted contexts. Automatic synonym acquisition experiments are conducted to evaluate the performance of the selection schemes mentioned above, reducing the unimportant contexts or co-occurrences. We use two evaluation measures: average precision (AP) and the correlation coefficient (CC). The three schemes are evaluated and compared in terms of performance gain/loss and computational cost reduction.

This chapter is organized as follows: in Section 6.2, context information (word-based context `wbc` and dependency-based context `dbc`) to which we paid attention in this chapter are introduced. In Sections 6.3 through 6.5, which is the key part of this chapter, three selection schemes are introduced. Each section accompanies experiment descriptions and results. Finally, Section 6.6 compares all these three schemes in terms of performance/cost trade-off. Section 6.7 concludes this chapter.

6.2 Contextual Information

In this section, how context types and co-occurrences are extracted from corpora before the selection is described in detail. We used two kinds of contextual information, namely, word-based context (`wbc`) and dependency-based context (`dbc`). We adopted these two kinds of contextual information because they are the most widely adopted ones for distributional similarity calculation.

The first and simplest context, *word-based context* (`wbc`) uses the words surrounding a target word as contexts. To capture this, we consider a window centered at the target word, and the tokens located within this window are extracted. The contexts for `wbc` in this chapter are represented as concatenations of relative positions and tokens. Take the following sentence for example:

> *The library has a large collection of classic books by such authors as Herrick and Shakespeare.*

The word "collection" has words such as "a," "large," "of" in its neighbor, so word-based context of "investigator" will be `L2:a`, `L1:large`, `R1:of`, and so on. The considered window is symmetric, and we adopted the maximum window radius of 3 in this chapter. We chose this setting because it is computationally expensive to consider windows with the radius larger than 3. Also, few researches which support their effectiveness can be found.

As the second, more sophisticated context, we used *dependency-based context* (`dep`) which is described in Section 2.2.

6.3 Category-based Selection

In this section, the simplest, category-based selection, where contexts are selected based on their category, is conducted and evaluated. We first set up the categories for `wbc` and `dbc`, and evaluate the synonym acquisition performance when individual categories are used as context.

6.3.1 Method

The context category is a subset of all the context types extracted from the corpus. The distributional similarity calculation is conducted based on each extracted category and its performance is compared. This operation corresponds to selecting a set of rows in the word-context co-occurrence matrix X in Section 4.2 at a time, as illustrated in Fig. 6.1.

The categories we set depend on the contextual information in use, which we will describe in the following. Notice that we can apply more sophisticated category selection methods, e.g., a finer-grained one incorporating richer kinds of syntactical information. However, we leave it to the other two selection schemes and in this section we conduct a somewhat naive selection as a generalization of previous ones such as [Hindle, 1990].

Word-based Context

For `wbc`, the context categories were set up according to the relative positions of context types. This means, in other words, context categories were the sets of context types covered by the windows with various ranges. We considered the following 15 window settings: [1, 0], [0, 1]; [2, 0], [1, 1], [0, 2]; [3, 0], [2, 1], [1, 2], [0, 3]; [3, 1], [2, 2], [1, 3]; [3, 2], [2, 3], and [3, 3], where the notation [l, r] represents the set of `wbc` extracted using the window covering l words on the left and r words on the right. Notice that all these 15 categories are the subsets of [3, 3], the case of using all the word-based contexts captured by the maximum window size on both sides.

Dependency-based Context

For `dbc`, the context categories were set up according to the grammatical relation (GR) of the context and the target word. We paid attention to the 10 most frequent GRs, i.e., `ncmod`, `dobj`, `ncsubj`, `obj`, `xmod` `cmod`, `ta`, `ccomp`, `obj2`, and `det`.

6.3.2 Experiment

Experimental Settings

The experimental settings are almost the same as the ones we used in Chapter 3. We used the 1994 New York Times articles, and extracted the word-based and dependency-based context. The cut-off frequencies were fixed to $\theta_w = \theta_c = 20$.

As for the weighting function and the similarity measure, `jaccard-s` and `pmi` were chosen considering the result of the preliminary experiment described in Chapter 3.

The weighting function PMI and the similarity measure Jaccard coefficient were chosen , where we compared the performance (average precision and correlation coefficient) of several weight functions such as tf, tf.idf, and PMI, and similarity measures such as cosine and Jaccard coefficient, while fixing other experimental parameters. PMI and Jaccard coefficient were also among the best performing measures described in [Curran and Moens, 2002a].

Computational Cost Estimation

When calculating the distributional similarity between two words w_i and w_j, two corresponding vectors $\mathbf{w}_i$ and $\mathbf{w}_j$ are constructed as described in Chapter 2 and all the elements of those vectors and the overlap between them needs to be assessed. This process takes a time proportional to the number of elements in those vectors, assuming that the vectors are sparsely represented, i.e., only non-zero elements are saved in the data structure. Thus, the computational cost of acquiring synonyms for a given query word is proportional to the number of the non-zero elements in the word-context co-occurrence matrix, because the similarity needs to be calculated for all the words $w_i, ..., w_N$. This roughly corresponds to the cost estimation of calculating the evaluation measures AP and CC, and most of other similarity-based tasks.

In the experiment, we therefore counted the number of the remaining co-occurrence types (not tokens), i.e., the remaining number of the matrix's non-zero elements after the selection is done, to evaluate the computational cost reduced by the selection. It is to note that the numbers of words and/or contexts are also the important factors that affect the computational cost. However, there may exist frequent words and context types (thus long vectors) and infrequent ones (thus short vectors). This makes it difficult to estimate the exact amount of calculation involved, using only the numbers of remaining words or context types. On the other hand, the number of the co-occurrences used in the distributional similarity calculation is almost proportional to the computational cost as explained above, which we will use as the rough cost estimates in the following experiments.

6.3.3 Result

Word-based Context

Table 6.1 shows the comparison of each `wbc` category, i.e., each window setting, and its AP and CC values. In the table, the categories are grouped by the total width of the window, and the rows are arranged by the order of *shift amount* from the center, from the left-most ones to the right-most ones.

Table 6.1: Result of category-based selection for `wbc`

Category	# contexts	# co-occurrences	AP	CC
[1, 0]	28,852	2,247,388	**0.1011**	**0.2513**
[0, 1]	17,646	1,866,524	0.0478	0.1894
[2, 0]	57,428	5,433,940	**0.1010**	0.2422
[1, 1]	46,498	4,113,912	**0.1063**	**0.2487**
[0, 2]	49,598	5,901,282	0.0603	0.2223
[3, 0]	88,488	9,606,758	0.0895	0.2465
[2, 1]	75,074	7,300,464	**0.1049**	0.2456
[1, 2]	78,450	8,148,670	0.0976	**0.2473**
[0, 3]	83,763	10,708,900	0.0641	**0.2573**
[3, 1]	106,134	11,473,282	0.0924	0.2427
[2, 2]	107,026	11,335,222	**0.0994**	0.1870
[1, 3]	112,615	12,956,288	0.0861	0.2405
[3, 2]	138,086	15,508,040	0.0904	0.2463
[2, 3]	141,191	16,142,840	0.0922	0.2393
[3, 3]	172,251	20,315,658	0.0846	**0.2511**

The first thing to note is that limiting the window length actually *increased* the performance in some cases. For example, even the [1, 0] window, where only the single word on the left is used as context, showed better performance compared to the full [3, 3] window. This is because most of the important words which act as strong clues for the semantic relatedness, such as adjectives and verbs of objects, are most likely to appear on the left side. The five highest AP and CC values of the 15 windows, which are emphasized in bold faces, suggest that the windows smaller than [3, 3] worked effectively enough. In most of these cases, both of the number of context types and co-occurrences, thus computational cost, are cut down by more than half, as shown in the second and third columns of the table. This result also suggests that windows shifted to the left, such as [1, 0] and [2, 0], show better performances in general. We suspect that this bias is related to the fact that the words having syntactical relations to the target words distribute mainly on the left in English. We do not go further into this topic here and will discuss it in another article.

It is to be noted that the performance of the categories differs between the evaluation measures, AP and CC. We suppose that this is because of the difference in characteristics of these two measures explained in Section 3.1. We'll come to this point later in Section 6.4.4.

Dependency-based Context

Table 6.2 shows the comparison of each `dbc` category based on GR and its AP and CC values. The categories are in the descending order of the number of co-occurrences belonging to the category.

Here we observe the strong correlation between the number of context types/co-

Table 6.2: Result of category-based selection for `dbc`

Category	# contexts	# co-occurrences	AP	CC
ncmod	49,541	2,966,778	**0.1094**	**0.2510**
ncsubj	6,942	1,179,344	**0.0686**	**0.2703**
dobj	3,975	870,377	**0.0905**	**0.2744**
ta	6,886	367,499	0.0374	0.0952
xmod	3,635	356,983	0.0500	0.1783
ccomp	2,269	261,559	0.0284	0.1765
cmod	2,515	224,334	0.0386	0.1721
det	4,540	202,786	0.0172	0.1116
obj	1,625	118,554	0.0598	0.2021
obj2	881	65,537	0.0178	0.0928

occurrences and the performance: the three highest values are all occupied by the three most frequent categories. Although this result re-confirmed Hagiwara et al.'s result [Hagiwara et al., 2006], which states that modification categories are strong evidence for distributional similarity, it is simply because the modification category is the most frequent grammatical relation, and the numbers of context types and co-occurrences contained in the category is an important factor. On the other hand, considering the data size and the performance, it is evident that `dobj` is on the whole a good-quality category because it performs comparable or even better compared to `ncmod` and `ncsubj`, with smaller numbers of co-occurrences (less than one thirds of `ncmod`, and three quarters of `ncsubj`). This also applies to `obj`, which demonstrates good performance with a very small number of data. This result gives a solid evidence which supports the use of widely adopted subject/object categories in the past, in terms of the trade-off between performance and computational cost (we call this *performance/cost trade-off* in the following).

These results of `wbc` and `dbc` suggest that even the simplest selection methods based on contextual categories work well for cost reduction, as well as performance improvements. We will discuss this performance/cost trade-off later in further detail, along with the results of other selection schemes.

6.4 Type-based Selection

We have just shown that even the simplest category-based selection worked well for boosting the performance as well as reducing the cost. However, it has some drawbacks such as the fact that the context categories must be given and fixed and it does not allow much finer selection, e.g., selecting effective context types one by one.

In this section, we propose the second selection scheme: type-based selection, where individual context types are scored, and selected or removed. This selection scheme corresponds to removing individual rows of the word-context co-occurrence matrix X one by one, as illustrated in Fig. 6.1, thus enabling finer-grained selection

compared to the category-based one mentioned above.

More specifically, type-based selection consists of these two steps: firstly, to each of context types, the *importance score* of the context is assigned using one of the measures described in the following. Secondly, the scored types are sorted by the importance score and ones with low values of importance scores are eliminated. The calculation of distributional similarity is performed using only the remaining rows. To calculate the importance scores for context types, we integrate the knowledge and result of feature selection methods proposed for text categorization or information retrieval, and adopt the five importance measures, namely, DF, TS, MI, IG, and CHI2, described in Section 6.4.2. Note that the measures except DF were first designed in order to select/remove features for *classification problems*, not for distributional similarity. Therefore, how to apply the rest of the measures to this distributional similarity problem is a big issue, which we deal with in the following section.

6.4.1 Formalization of Distributional Similarity

To apply the pair-based importance measures, we formalize distributional similarity as a "pair classification" problem as described below.

First of all, we deal with word pairs, instead of individual words, as the instances of classification, and define features $f_1, ..., f_m$ corresponding to context types $c_1, ..., c_m$. The feature value is defined $f_j = 1$ if the two words of the pair have the context c_j in common, and $f_j = 0$ otherwise. Then, we define the target class s, so that $s = 1$ when the pair is semantically related, and $s = 0$ if not. With these defined, distributional similarity is formalized as a binary, pair classification problem which assigns the word pairs to the classes $s \in \{0, 1\}$ based on the features $f_1, ..., f_m$. Finally, to calculate the specific values of the following context importance scores, we prepare two test sets of related word pairs for class $s = 1$ and unrelated ones for class $s = 0$. This enables us to apply existing feature selection methods designed for classification problems to the automatic context selection.

The two test sets, related and unrelated one, are prepared using the *reference sets* described in Section 3.1. More specifically, we created 5,000 related word pairs by extracting from synonym pairs in the reference set, and 5,000 unrelated ones by firstly creating random pairs of the LDV words, whose detail is described later, then manually making sure that no related pairs are included in these random pairs.

6.4.2 Method

In the following, the importance score measures, namely, DF, TS, MI, IG, and CHI2, are introduced. Remember that n and m represent the number of word and context types, respectively, and $N(w, c)$ denotes the frequency count of co-occurrence of word w and context c.

Document Frequency (DF)

Document frequency (DF), commonly used for weighting in information retrieval, is the number of documents a term co-occurs with. However, in the distributional

similarity settings, DF corresponds to *word frequency*, i.e., the number of unique words the context type co-occurs with:

$$df(c) = |\{w|N(w,c) > 0\}|. \tag{6.1}$$

The motivation behind adopting DF as a context selection criterion is the assumption that the contexts shared by many words should be informative. It is to note, however, this concept that higher DF values corresponds to higher importance is totally contrary to the underlying assumption of idf (inverse document frequency) that the context types with too high DF, i.e., so-called *stopwords*, are not useful. On the other hand, the context types with too low DF values are undoubtedly unimportant, because it is very rare that such types are shared by pairs of words and act as useful features for pairs. We suppose that this DF measure can roughly capture this tendency.

Term Strength (TS)

Term strength (TS), proposed by Wilbur and Sirotkin [Wilbur and Sirotkin, 1992] and applied to text categorization by Yang and Wilbur [Yang and Wilbur, 1996], measures how likely a term is to appear in "similar documents," and it is shown to achieve a successful outcome in reducing the amount of vocabulary for text retrieval. For the distributional similarity setting, TS is defined as:

$$ts(c) = P(c \in C(w_2)|c \in C(w_1)), \tag{6.2}$$

where (w_1, w_2) is a related word pair and $C(w)$ is the set of context types co-occurring with the word w, i.e., $C(w) = \{c|N(w,c) > 0\}$. Note that the value of $ts(c)$ is symmetric because we consider the similarity relation to be symmetric, i.e., when (w_1, w_2) is a synonymous word pair, (w_2, w_1) is also synonymous. The value of $ts(c)$ is calculated, letting P_H be a set of related word pairs, as

$$ts(c) = \frac{|\{(w_1, w_2) \in P_H|c \in C(w_1) \cap C(w_2)\}|}{|\{(w_1, w_2) \in P_H|c \in C(w_1)\}|}. \tag{6.3}$$

What makes TS different from DF is that it requires a reference set P_H consisting of related word pairs. We used the reference set for class $s = 1$ as P_H described in Section 6.4.1.

Mutual Information (MI)

Mutual information (MI), commonly used for word association and co-occurrence weighting in statistical NLP, is the measure of the degree of dependence between two events. The *pointwise* MI value of feature f and class s is calculated as:

$$I(f,s) = \log \frac{P(f,s)}{P(f)P(s)}. \tag{6.4}$$

To obtain the final context importance, we combine the MI value over both of the classes as $I_{\max}(c_j) = \max_{s\in\{0,1\}} I(f_j, s)$. Note that, here we employed the maximum value of pointwise MI values since it is claimed to be the best in [Yang and Pedersen, 1997], although there can be other combinations such as weighted average.

Information Gain (IG)

Information gain (IG), often employed in the machine learning field as a criterion for feature importance, is the amount of gained information of an event by knowing the outcome of the other event. IG is calculated as the weighted sum of the pointwise MI values over all the event combinations:

$$G(c_j) = \sum_{f_j \in \{0,1\}} \sum_{s \in \{0,1\}} P(f_j, s) \log \frac{P(f_j, s)}{P(f_j)P(s)}. \tag{6.5}$$

χ^2 Statistic (CHI2)

χ^2 statistic (CHI2) estimates the lack of independence between classes and features, which is equal to the summed difference of observed and expected frequency over the contingency table cells. More specifically, letting $F_{nm}(n, m \in \{0, 1\})$ be the number of word pairs with $f_j = n$ and $s = m$, and the number of all pairs be N, χ^2 statistic is defined as:

$$\chi^2(c_j) = \frac{N(F_{11}F_{00} - F_{01}F_{10})^2}{(F_{11} + F_{01})(F_{10} + F_{00})(F_{11} + F_{10})(F_{01} + F_{00})}. \tag{6.6}$$

6.4.3 Experiment

In this experiment, the context types are sorted by the importance scores mentioned above and ones with the lowest scores were removed in order. We started the selection with all the 172,251 types for `wbc` and 83,029 types for `dbc`, i.e., 100% of the originally extracted context types, reducing the percentage of remaining types until only 0.2% (344 for `wbc` and 166 for `dbc`) were left. The evaluation measures, AP and CC, were calculated at each step.

The conditions such as the corpus, parser (RASP2), contextual information (`dbc`, `wbc`), frequency cut-off, are exactly the same as the previous experiment (Section 6.3.2).

6.4.4 Result

The performance change when the context types of `wbc` were selected/removed is plotted in Fig. 6.2. The overall observation is that the performance not only kept the original level but also slightly improved even during the "aggressive" reduction when more than 80% of the original types were eliminated and around 30,000 context types were left. It was not until 90% (approx. 17,000 remaining) elimination that the AP values began to fall. The usefulness of context selection was even more significant for CC, where the value increased more than 20% during the 90% reduction. The reason why this performance increase was observed is that, the context types to which lower importance scores were assigned were actually "noise" which may have hindered the accurate distributional similarity calculation and they could be removed by the context selection technique proposed here.

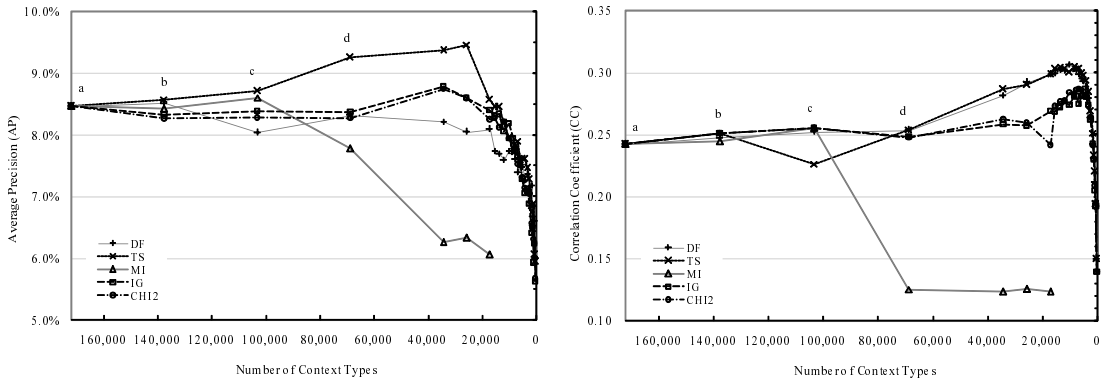

Figure 6.2: Performance change v.s. context types on type-based context selection for `wbc`

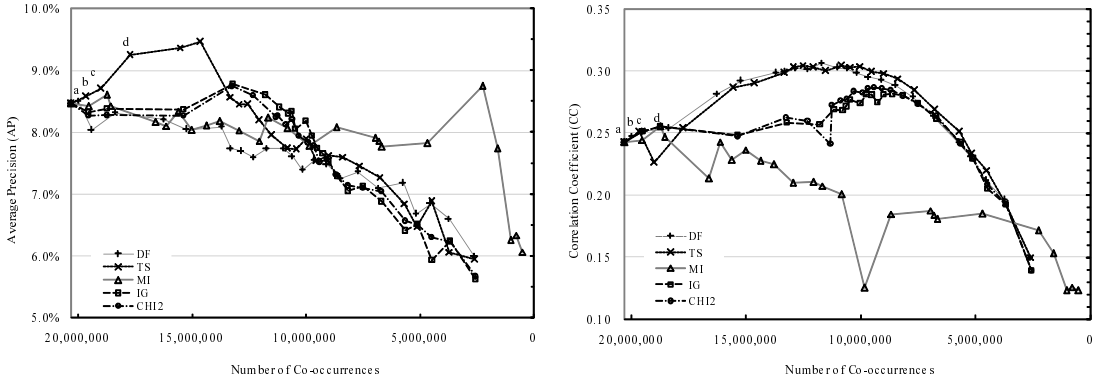

Figure 6.3: Performance change v.s. co-occurrences on type-based context selection for `wbc`

We observed a slight difference regarding which of the five measures were effective. More specifically, TS, IG and CHI2 worked well for AP, while DF and TS did for CC. On the whole, TS was performing the best, with IG and CHI2 coming next, whereas the performance of MI quickly worsened. Although the task was different, this experiment showed a very consistent result compared with the one of Yang and Pedersen's [Yang and Pedersen, 1997]. This means that feature selection methods are also effective for context selection in distributional similarity. Our formalization of the problem described in Section 6.4.1 turned out to be appropriate for the purpose.

In order to clarify the trade-off between the synonym acquisition performance and computational cost, we re-plotted the result of the current experiment by taking the number of co-occurrences, i.e., the number of non-zero elements in the word-context co-occurrence matrix, which remained after the selection operation, as x-axis. The result is displayed in Fig. 6.3, and it reveals an interesting surprise.

The first thing to note is that the decline curve of AP v.s. co-occurrences is more pronounced, compared to Fig. 6.2, making it almost linear. This difference in shape

arises from the fact that most *co-occurrences* remain intact during the plateau shape (from 172,251 types to approx. 80,000 types) of Fig. 6.2, where context *types* are aggressively eliminated but the performance keeps almost unchanged. To exemplify this, four of the corresponding points in Figs. 6.2 and 6.3 are marked by letters "a," "b,", "c," and "d." This is because the measures introduced above, except for MI, explicitly or implicitly favor frequent context types, thus type-based context selection using these four measures removes infrequent types first, keeping as many co-occurrence as possible. The CC curve also shows a quite different shape, where the peak located at the far right of Fig. 6.2 is now shifted toward the center, meaning that the best CC is achieved when about a half of the co-occurrences are eliminated.

On the other hand, MI, which we had been considering to be the worst-performing importance measure in terms of performance/type trade-off, shows quite different characteristics and now it tops the other four measures and demonstrates the best performance/cost trade-off of five, when only 10% of the original co-occurrences are selected. This makes MI the best candidate for the applications which poses an extremely high demand for computational cost, such as machine learning techniques and clustering. Other four measures showed little difference among them between Figs. 6.2 and 6.3, with their order slightly different.

The above observation indicates that, the best measures in terms of performance/types trade-off, such as TS and CHI2, do not always show the best performance/cost trade-off, which can be achieved by MI. The choice is up to the tasks and requirements in practice — the former measures would be appropriate for the tasks which press high demands for low dimensionality of feature spaces, whereas the latter measure can be a good candidate for the tasks dealing with sparse matrices and vectors.

Here we notice that the results are slightly different for AP and CC, and would like to discuss the cause below. In category-based selection, AP values declined greatly for [0, 1] and [0, 2] of `wbc`, and `obj` of `dbc`, while the change of CC values was relatively small, as shown in Tables 6.1 and 6.2. Furthermore, the peaks of CC values shown in Figs. 6.2 and 6.4 were shifted towards fewer context types when compared to AP values. What these suggest is, in order to accurately calculate AP, i.e., to accurately acquire synonyms, we need more context types because the overlap of vectors has to be precisely obtained, whereas in order to accurately calculate CC, i.e., to accurately assign similarity, we do not need as many context types because the non-overlapping elements of vectors are also important. This is intuitively correct — most of elements in vectors do not overlap because there are so many types and a lot of context types should remain in order to obtain sufficient overlaps.

Almost the same trend was observed for both `wbc` and `dbc`, with the result shown in Figs. 6.4 and 6.5. From the results described in this section, we can conclude this type-based context selection is general enough to be applied for any distributional similarity settings.

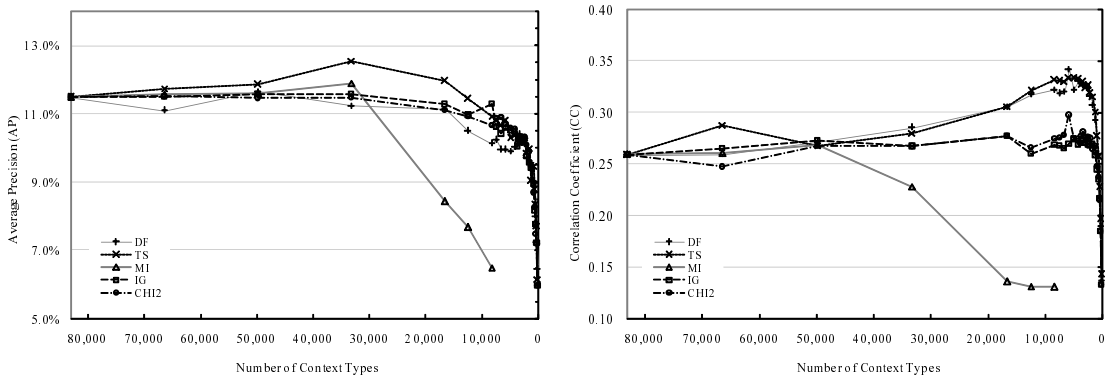

Figure 6.4: Performance change v.s. context types on type-based context selection for `dbc`

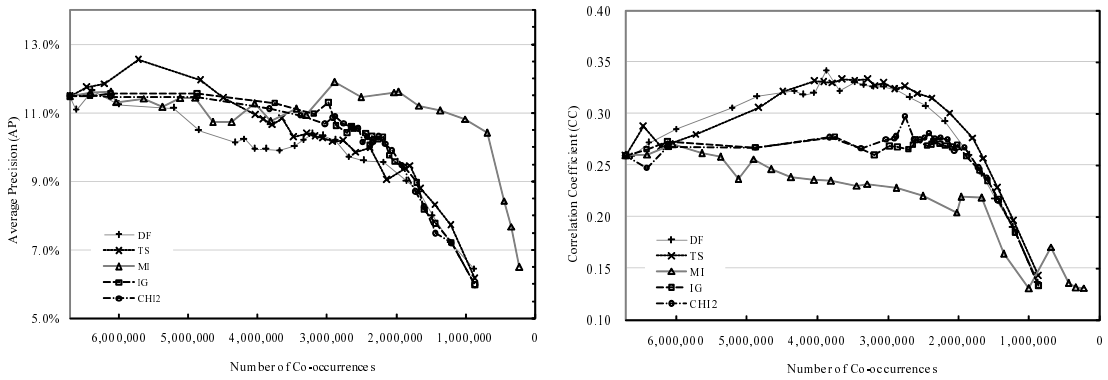

Figure 6.5: Performance change v.s. co-occurrences on type-based context selection for `dbc`

6.5 Co-occurrence based Selection

We have previously shown that type-based selection could provide a general-purpose framework for feature space reduction. However, even within a single context type, the importance of individual *co-occurrence* may vary according to the word with which the context co-occurs. Thus finer-grained selection than type-based one is appropriate in some cases. Now let us consider a situation where a single context type c co-occurs with a very frequent word w_1 and an infrequent word w_2. In this case, the co-occurrence (w_1, c) can be ignored without worsening the performance because w_1 may have co-occurrences with other context types which are discriminative enough to precisely characterize the word w_1. The same will not be true for w_2 — the co-occurrence (w_2, c) is one of the scarce features w_2 has. As such, it should not be lightly weighted, let alone ignored.

This leads us to the third and finest-grained context selection scheme, *co-occurrence based selection*, where individual co-occurrences (w, c) are weighted, sorted, and removed. This selection operation corresponds to removing (i.e., replacing with zeros) individual elements of the word-context co-occurrence matrix X one by one, as illustrated in Fig. 6.1, thus enabling the finest-grained selection of the three schemes proposed in this chapter.

6.5.1 Method

Co-occurrence based selection works quite similarly to type-based selection — the importance score is assigned to every co-occurrence (w, c), instead of context type c, and the co-occurrences are sorted by the scores and ones with low importance are eliminated. The calculation of distributional similarity is performed using only the remaining elements. To calculate the importance score, here we pay attention to these four weight functions listed below: `tfidf`, `pmi`, `chi2`, and `ttest`.

tf.idf (term frequency, inverse document frequency) Weighting (tfidf)

As one of the traditional yet commonly adopted weighting measures, `tfidf` is widely used for information retrieval systems. The raw co-occurrence frequency, tf, is weighted by idf:

$$\mathrm{wgt}(w, c) = N(w, c) \cdot \mathrm{idf}(c), \quad \mathrm{idf}(c) = \log \frac{n}{\mathrm{df}(c)}, \tag{6.7}$$

where $\mathrm{df}(c)$ is the number of word types with which the context type c co-occurs, i.e., $\mathrm{df}(c) = |\{w | N(w, c) > 0\}|$. In the experiments, the idf values are normalized so that the maximum equals to 1.

Pointwise Mutual Information (pmi)

The amount of information that the appearances of w and c have in common is defined as their pointwise mutual information (`pmi`):

$$\mathrm{wgt}(w, c) = \log \frac{p(w, c)}{p(w)p(c)}, \tag{6.8}$$

where the probabilities are calculated empirically, e.g. $p(w,c) = N(w,c)/\sum_{w',c'} N(w',c')$. Note that this `pmi` is also used to weight co-occurrences throughout this chapter.

$\chi 2$ statistic (chi2)

χ^2 statistic (`chi2`), which was also used for type-based selection, is used to measure the lack of dependence between w and c. χ^s is defined as:

$$\mathrm{wgt}(w,c) = \frac{K(N_{11}N_{00} - N_{01}N_{10})^2}{(N_{11}+N_{01})(N_{10}+N_{00})(N_{11}+N_{10})(N_{01}+N_{00})}, \tag{6.9}$$

where N_{11}, N_{10}, N_{01}, and N_{00} are the frequency counts of w co-occurring with c, w without c, c without w, and neither, respectively. K is the total number of co-occurrences, i.e, $K = N_{11} + N_{10} + N_{01} + N_{00}$.

t-statistic (ttest)

t-statistic, usually used to test whether the appearances of word w and context c are statistically independent, is used as a weight function:

$$\mathrm{wgt}(w,c) = \frac{p(w,c) - p(w)p(c)}{\sqrt{p(w)p(c)}}. \tag{6.10}$$

Whereas t-statistic is usually employed to discover collocations from corpora, Curran and Moens [Curran and Moens, 2002a] proposed using it as a weight function and showed that it achieved higher performance compared to the others.

6.5.2 Experiment

In this experiment, co-occurrences are sorted by the importance scores mentioned above and ones with the lowest scores were removed in order. We started the selection with all 20,315,658 co-occurrence types for `wbc` and 6,710,937 co-occurrence types for `dbc`, i.e., 100% of the originally extracted co-occurrences, reducing the percentage of remaining ones until 2.5% (507,891 for `wbc` and 167,773 for `dbc`) were left. The evaluation measures, AP and CC, were calculated at each step.

The conditions such as the corpus, parser (RASP2), contextual information (`dbc`, `wbc`), frequency cut-off, are exactly the same as the previous experiment (Section 6.4.3).

6.5.3 Result

The performance change on co-occurrence based selection applied for `wbc` is displayed in Fig. 6.6. We observed not only the significant tolerance of AP to a small number of selected co-occurrence, but also much greater increase of AP, when compared to type-based selection shown in Fig. 6.3. At the peak, which was seen around 5,000,000 co-occurrences, i.e., less than quarter of the original data size, the increase of AP was approx. 25% percent of that of before the selection.

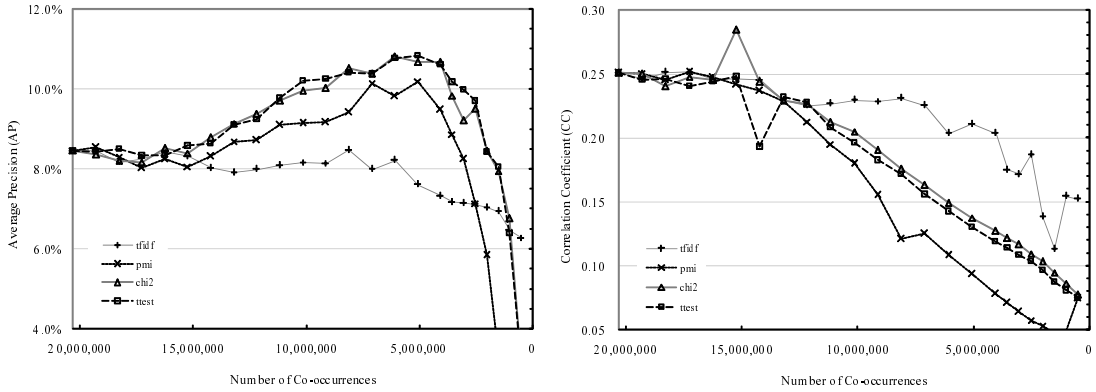

Figure 6.6: Performance change on co-occurrence based context selection for `wbc`

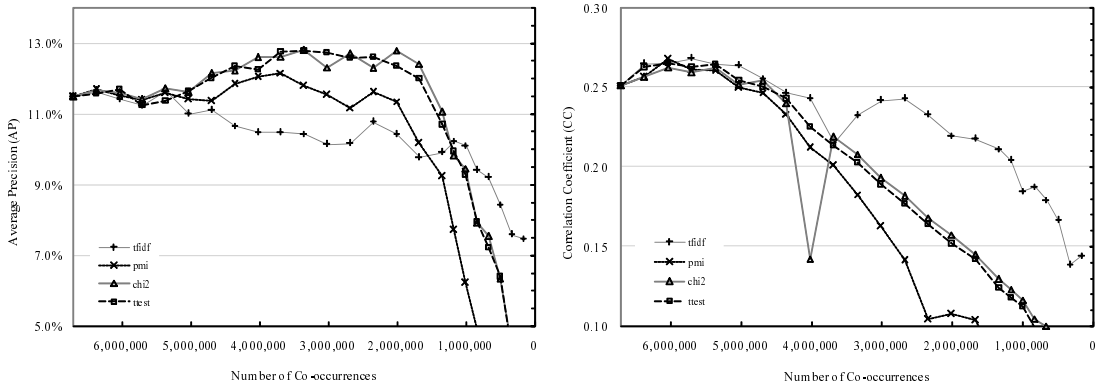

Figure 6.7: Performance change on co-occurrence based context selection for `dbc`

On the other hand, CC declined almost linearly as the co-occurrences were selected for most of the importance measures. The decline was steeper than type-based selection. This result is somewhat contrary to the previous result of type-based selection, where CC showed the tolerance to fewer number of co-occurrences. As far as we currently speculate, it is because CC calculation requires a lot of co-occurrences, most of which are contained in the non-overlapping elements of vectors, and they were removed in the early stage of co-occurrence based selection. In contrast, AP calculation requires many overlapping context types as we mentioned in Section 6.4.4, although this overlap is relatively small in terms of the number of co-occurrences. A large part of this remains even after co-occurrence selection, which boosts the AP values.

As for the importance measures, `chi2` and `ttest` performed best of all, while `tfidf` requires special attention. Although it did not work well for "moderate" selection where around half of the co-occurrences were removed, it outperformed for the "aggressive" selection where less than quarter of the co-occurrences remained, this trend being especially outstanding for CC. Different importance measures exhibit dif-

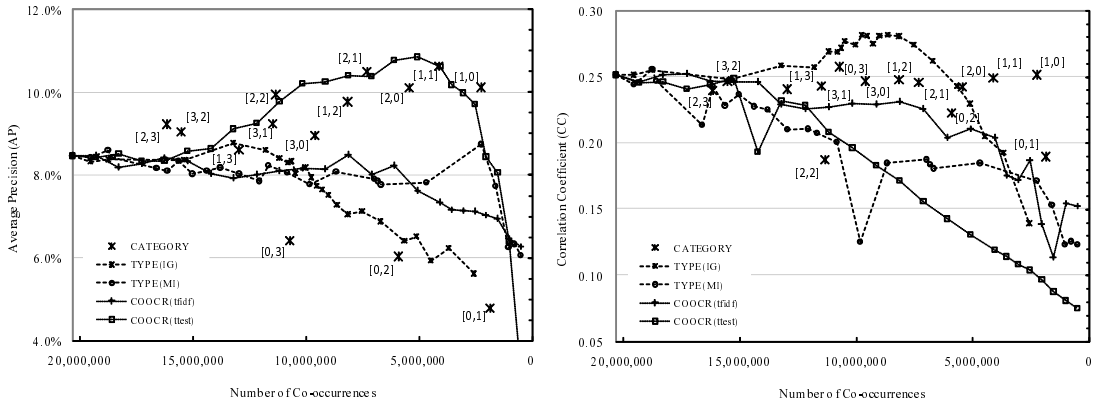

Figure 6.8: Comparison of performance-cost trade-off of three schemes for `wbc`

ferent characteristics, and the choice depends on how aggressive reduction the task demands. The result for `dbc` also reveals the same tendency as shown in Fig. 6.7. We suppose that the effectiveness of co-occurrence based selection, as well as the characteristics of individual importance measures, is general enough and applicable to other kinds of settings.

6.6 Comparison of Three Selection Schemes

Finally, we compare all the three selection schemes proposed in this chapter in terms of performance and computational cost. The results of category-based selection (`CATEGORY`), two representative measures from type-based selection (`TYPE (IG)` and `TYPE (MI)`), and from co-occurrence based selection (`COOCR (tfidf)` and `COOCR (ttest)`) are plotted in a single plane (Fig. 6.8). The result of `CATEGORY` is represented by the labelled discrete points corresponding to individual categories. The graph takes the number of co-occurrences remained after the selection as x-axis and the performance as y-axis, thus we can compare the performance/cost trade-off of the three schemes.

The graph reveals that, on the whole, `COOCR (ttest)` works well for AP and `TYPE (IG)` does for CC. Although this result looks somewhat contradictory, the difference in the characteristics of the evaluation measures explains this — recall that type-based selection helps keep a high CC level, while co-occurrence based selection keeps a high AP level, which we discussed in Sections 6.4.4 and 6.5.3. Thus selection methods should be carefully chosen depending on the nature of actual tasks to which one is applying these methods, considering the characteristics of AP and CC as described in Section 5.

Considering both evaluation measures, `TYPE (IG)` will be the choice for "moderate" selection which removes the co-occurrences up to approx. 50%, and `TYPE (MI)` for "aggressive" selection, where less than 20% are left. A little surprising fact is that we cannot miss the effectiveness of the simplest, category-based context selec-

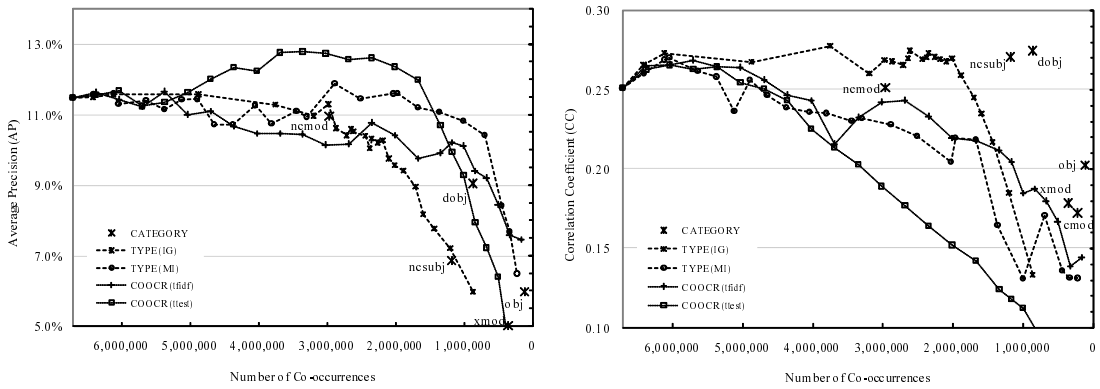

Figure 6.9: Comparison of performance-cost trade-off of three schemes for `dbc`

tion. Actually, the windows [1,0], [1,1], [2,0], [2,1], corresponding to four points found at upper right of the graph, are almost the best of all three schemes, in terms of performance/cost trade-off.

However, other `wbc` categories, e.g., [3,1], [3,0], and [1,2], still cannot beat `COOCR(tfidf)` or `TYPE(IG)`. This suggests that the effectiveness of `CATEGORY` is only observed for rather limited conditions such as [1, 0]. After all, it only worked well for types of context (`wbc` and `dbc`) where categories can be clearly defined, and under no conditions for computational cost or dimensionality. It is also difficult for type-based selection to predict the computational cost and the performances we would obtain after the selection.

This implies that type-based and co-occurrence based selection methods still have these advantages. First, they are general enough to be applied to context where categories cannot be clearly defined. Category-based selection may have difficulties when applied to newly proposed types of context such as dependency path [Lin and Pantel, 2001], while the other two methods can be easily adopted for them. Secondly, they are flexible enough to meet the imposed conditions in terms of computational cost and dimensionality. In some tasks such as machine learning, computational resources are demanding and some conditions might be already given beforehand in terms of computational cost and dimensionality. Type-based and co-occurrence based selection can meet them by adjusting the "aggressiveness" of selection flexibly. Thirdly, it is easy to predict the computational cost and the performance achieved after type-based or co-occurrence based selection is applied, but not for category-based selection.

To sum up, in some limited cases category-based selection works well, while in most cases one can benefit from the generality and flexibility of the other two selection methods.

A similar comparison is conducted for `dbc`, and the result is shown in Fig. 6.9. The characteristics of the three schemes are almost the same, and again, the relative effectiveness of `CATEGORY` is clear, though not as outstanding as `wbc`. The best category is `dobj`. This evidence again supports the use of the combination of subject and

object relations as context, as done in many previous works.

Lastly but not least importantly, the three schemes are not mutually exclusive but freely combined and used. For example, we can firstly apply category-based selection using `obj`, then type-based selection using `DF`, and finally the remaining co-occurrences can be further thinned out using co-occurrence based selection with `pmi`. Although we do not investigate the impact of this combination, one can choose any schemes as well as measures, depending on the demand for the semantic space dimensionality and computational cost.

6.7 Conclusion

In this chapter, we proposed three schemes of context selection for distributional similarity: category-, type- and co-occurrence based selection. Category-based selection is the simplest selection where context types are selected/removed based on the syntactical categories the types belong to. Finer-grained, type-based selection enables the selection of context types one by one, by individually assigning them importance scores. For this purpose, we reformalized the distributional similarity problem as a classification problem, enabling the application of existing measures for feature selection to the current task. The finest, co-occurrence based selection selects or removes individual co-occurrences of word and context, assigning them importance scores.

In the experiments, we investigated how well the three selection schemes, as well as individual importance measures, work for removing unwanted contexts for word-based context `wbc` and dependency-based context `dbc`. Automatic synonym acquisition was used as a task to evaluate the performance, and average precision (AP) and correlation coefficient (CC) were employed as evaluation measures.

In the first experiment, we compared the performance and context reduction amount (the numbers of remaining context types and co-occurrence) for category-based selection. It showed that limiting the categories actually increased the performance, and appropriate choice of categories, such as [1,0]-window for `wbc` and `dobj` for `dbc`, were indeed effective for both performance and computational cost.

The second experiment, where the performance change on type-based context selection was evaluated, showed that the performance not only kept the original level but also slightly improved even during the "aggressive" reduction when more than 80% of the original types were eliminated. This implies the effectiveness of our formalization of distributional similarity problem for this task. As for the importance measures, TS, IG and CHI2 were showing one of the best performance/type trade-off, while MI was the best in terms of performance/cost trade-off.

The third experiment revealed that co-occurrence based selection demonstrated even better performance/cost trade-off in some cases, especially when `chi2` or `ttest` were used.

Finally, we compared above three selection schemes and clarified the characteristics of the schemes and the measures. It also showed the effectiveness of the simplest category-based selection. This result supports the use of subject and object relations as context, as done in many previous works. At the same time, type-based

and co-occurrence based selection methods also work for both `wbc` and `dbc`, showing that these method can be generally and flexibly used for any kinds of context and dimensionality/computational cost constraints.

In this chapter, we only presented a considerably simple classification model of distributional similarity to assign importance scores for distributional similarity. However, much more complex and advanced models can be considered, whose effectiveness for type-based context selection should be investigated in the future.

Finally, the selection schemes we presented all assume the mutual independence of context types and co-occurrences, although this assumption is often unrealistic. Some feature selection methods incorporating mutual correlation among features are proposed for machine learning [Ding and Peng, 2003]. The same technique may be applicable for context selection as well.

Chapter 7

Supervised Synonym Acquisition

7.1 Introduction

Although distributional similarity-based methods have been successful in acquiring related words, this involves various parameters such as similarity measures and weight functions, as we've seen in Chapter 2. As Weeds et al. [Weeds et al., 2004] pointed out, "it is not at all obvious that one universally best measure exists for all application," thus the parameters must be tuned by hand in an ad-hoc manner, depending on the task. Several researchers [Lee, 1999, Curran and Moens, 2002a, Weeds et al., 2004] have compared a number of weight functions and similarity measures, although they ended up with varied results. The fact that no theoretic basis is given to the distributional similarity setting is making the matter more difficult.

On the other hand, if we pay attention to lexical knowledge acquisition in general, a variety of systems which utilized *syntactic patterns* are found in the literature. In her landmark paper in the field, Hearst [Hearst, 1993] utilized syntactic patterns such as "such X as Y" and "Y and other X," and extracted the hypernym/hyponym relation of X and Y. Riloff and Shepherd [Riloff and Shepherd, 1997] and Roark and Charniak [Roark and Charniak, 1998] applied this idea to extraction of words which belong to the same categories, utilizing word co-occurrence in a context window or syntactic relations such as conjunctions and appositives. The advantage of syntactic patterns is that they can easily converted to features, and machine learning approach can be used to discriminate the relations that the words have. Snow et al. [Snow et al., 2004], for example, took this approach and built hypernym classifiers based on dependency patterns.

Some researchers have started to use these two independent approaches, distributional similarity and syntactic patterns together. Lin et al. [Lin et al., 2003] first collected distributionally similar words, i.e., related words, then filtered out antonyms and cohyponyms (words that share the same hypernym) using syntactic patterns such as "from X to Y" and "either X or Y." This approach partially solved the problem that distributionally similar words tend to include not only synonyms but also antonyms and cohyponyms. Mirkin et al. [Mirkin et al., 2006] tried to integrate these two approaches, by using the distributional similarity as a feature along with other

pattern-based features, and constructing a classifier to detect lexical entailment relations. Although they reported that their classification-based system successfully improved the performance, it only achieves partial integration and was still relying on an independent module to compute the similarity. This configuration inherits a large portion of drawbacks of the similarity-based approaches mentioned above. To achieve full integration of both approaches, we suppose that re-formalization of the similarity-based approach would be essential, as pattern-based approach is enhanced with the supervised machine learning.

In this chapter, we propose a novel approach to automatic synonym identification based on supervised learning technique which does not use any kind of predetermined similarity measures. The contribution of this chapter is four-fold. Firstly, we re-formalize synonym acquisition as a classification problem: one which classifies *word pairs* into synonym/non-synonym classes, without depending on a single value of distributional similarity. In this way, the synonym classification can benefit from various machine learning techniques including the kernel trick. Secondly, in addition to conventional *common features*, we propose a new set of features, i.e., *distributional features*, which correspond to the degree of commonality of individual context types shared by word pairs. We show in the experiments their effectiveness in classifying synonym pairs. Thirdly, we propose fully-integrated synonym classifiers based on both common/distributional features and *pattern-based features*, which have also been used for lexical knowledge acquisition tasks. We build eight classifiers based on distributional, common, and/or pattern-based features. In the experiments, their performances are compared in terms of precision and recall of synonym acquisition, and the differences of actually acquired synonyms are to be clarified. Fourthly, instead of training a classifier, we directly learn the similarity/distance measure itself as a different approach. Given the known instances of similar or dissimilar words, we estimate the parameters of the Mahalanobis distance. We compare a number of measures in the experiments.

The rest of this chapter is organized as follows: in Section 7.2, we propose the classification-based approach to synonym acquisition, and the construction details of common and distributional features are described. The other type of features, pattern-based features, are defined in the following Section 7.3. We build eight types of synonym classifiers, and compare their performances in Section 7.4. Section 7.5 describes the metric-learning approach to synonym acquisition and its performance results. Section 7.6 concludes this chapter.

7.2 Pairwise Classification

This section describes a different approach to synonym acquisition, i.e., pairwise synonym classification. We adopt two schemes to construct features in the classification task: *common features* and *distributional features*. The detail is given below.

7.2.1 Common Features

As described above, the conventional distributional similarity is an *unsupervised* model heavily dependent on weighting functions and similarity/distance measures, which must be carefully hand-tuned because the choice directly and considerably affects the performance. To circumvent this, in this section we re-formalize the synonym acquisition as a *supervised*, pairwise classification model which does not depend on conventional similarity measures.

In the distributional similarity model, the similarity is computed for a pair of words, say, x and z, based on the features assigned to individual words. We call these features *word features* in the following. The pairwise classification model, on the other hand, directly deals with word pairs, say, $p = (x, z)$, as the target to classify. The feature values are computed from word features of the pair words, and directly assigned to the pairs. We call these features *pair features*, as opposed to word features. In the following, we suppose that each word is represented as a vector of word features as:

$$\mathbf{x} = [x_1\ x_2\ \ldots\ x_M]^T, \quad x_j = \mathrm{wgt}(x, c_j), \tag{7.1}$$

$$\mathbf{z} = [z_1\ z_2\ \ldots\ z_M]^T, \quad z_j = \mathrm{wgt}(z, c_j). \tag{7.2}$$

This pairwise classification approach was used for several lexical relation identification task such as [Bilenko and Mooney, 2003, Mirkin et al., 2006], but not for synonym acquisition so far to the best of authors' knowledge.

The first one of the two feature construction methods we adopted is the one which constructs pair features by taking the simple products of common word features, as usually done in pairwise classification tasks [Bilenko and Mooney, 2003]. This scheme constructs the featurev ector $\mathbf{p}_{\mathrm{cfeat}}$ for the pair $p = (x, z)$ as:

$$f_j^C(x, z) = \mathrm{wgt}(x, c_j) \cdot \mathrm{wgt}(z, c_j) \tag{7.3}$$

$$\mathbf{p}_{\mathrm{cfeat}} = [f_1^C(x, z)\ f_2^C(x, z)\ \ldots\ f_M^C(x, z)]^T. \tag{7.4}$$

This feature construction can be interpreted as follows — instead of using a single, integrated value of the summation in the cosine similarity measure of Equation (2.8), the summation is expanded and all the products in the summation are used as individual features, except for the normalization factors in the denominator. In this way, we can expect that appropriate weights are optimally assigned by learning algorithms and contributing features would be encouraged, increasing the acquisition performance.

7.2.2 Distributional Features

The pair features do not have to be constructed from individual word features, but directly from co-occurrence probabilities obtained from corpora. The other scheme we propose here is to directly assign pair features, which we call *distributional features*, to the pair p as:

$$\mathbf{p}_{\mathrm{dfeat}} = [f_1^D(x, z)\ f_2^D(x, z)\ \ldots\ f_M^D(x, z)]^T \tag{7.5}$$

The value of distributional features $f_j^D(x, z)$ is determined so that it represents the degree of commonality of context c_j shared by the word pair (x, z). *Pointwise total correlation*, one of the generalizations of pointwise mutual information, can measure this information amount. We adopt this as the feature value:

$$f_j^D(x, z) = \log \frac{P(x, z, c_j)}{P(x)P(z)P(c_j)}. \tag{7.6}$$

The advantage of this feature construction is that, given the independence assumption between word x and z, the feature value is easily calculated as the simple sum of two corresponding pointwise mutual information weights as:

$$f_j^D(x, z) = \mathrm{PMI}(x, c_j) + \mathrm{PMI}(z, c_j), \tag{7.7}$$

where the value of $\mathrm{PMI}(x, c_j)$, which is also the weights $\mathrm{wgt}(x, c_j)$ assigned for distributional similarity, is calculated as shown in Equation (2.25). Note that the feature value $f_j^D(x, z)$ in Equation (7.6) is finite only when the context c_j co-occurs with both x and z. In practice, however, we used the definition in Equation (7.7) with the independence assumption, and the value of $f_j^D(x, z)$ is finite even for the case where the context c_j does not co-occur with both x and z, i.e., $\mathrm{PMI}(x, c_j) = 0$ or $\mathrm{PMI}(z, c_j) = 0$.

Notice that, as long as PMI is used as the weighting functions, these two feature construction schemes differ only in that the former uses the products, as shown in Equation (7.3), while the latter uses the sum. Therefore, when the weights are PMI, Equation (7.5) can be rewritten as:

$$\mathbf{p}_{\mathrm{dfeat}} = [x_1 + z_1 \quad x_2 + z_2 \quad \ldots \quad x_M + z_M]^T. \tag{7.8}$$

In the experiments, we compare the performance of common features and distributional features.

7.3 Pattern-based Features

This section describes how to extract and construct the other type of features — syntactic patterns extracted from sentences.

7.3.1 Syntactic Pattern Extraction

Syntactic patterns are usually simple string patterns with word slots, such as "X such as Y" and "Y and other X," which directly express the relation between two words. For lexical relation extraction, such patterns are often converted into web search queries and issued to web search engines to obtain the occurrence count of the patterns, as done in [Hearst, 1993, Lin et al., 2003, Mirkin et al., 2006] and [Pantel and Pennacchiotti, 2006]. Although such string-based patterns are simple to implement and accurate, they often lack robustness and generality because linguistic variations and noise in the sentence may prevent accurate matching of those patterns.

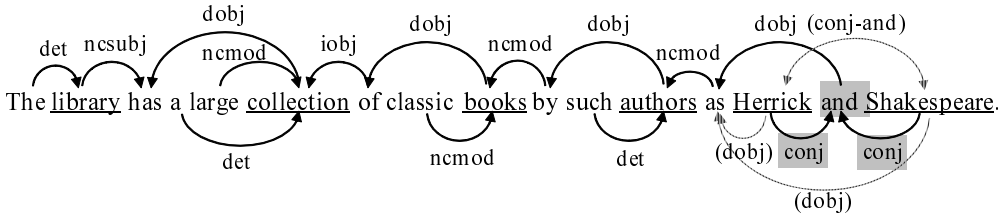

Figure 7.1: Dependency structure of the example sentence, along with conjunction shortcuts (dotted lines).

This problem can be addressed by considering richer definition of syntactic patterns, i.e., ones based on dependency structure of sentences. Following Snow et al.'s definition [Snow et al., 2004], in this chapter we define the syntactic pattern of words w_1, w_2 as the concatenation of the words and relation labels which are located on the dependency path from w_1 to w_2, not including w_1 and w_2 themselves.

For example, the syntactic pattern of word *authors* and *books* in Figure 7.1 is `dobj:by:ncmod`, while that of *authors* and *Herrick* is `ncmod-of:as:dobj-of:and:conj-of`. Notice that, although not shown in the figure, every relation edge has a reverse edge as its counterpart, with the direction opposite and the postfix "`-of`" attached to the label. For example, the `dobj` edge which goes from *collection* to *has* has its reverse, `dobj-of`, which goes from *has* to *collection*. This allows to follow the relations in reverse, increasing the flexibility and expressive power of patterns.

In the experiment, we limited the maximum length of syntactic path to five, meaning that word pairs having six or more relations in between were disregarded. Also, we considered *conjunction shortcuts* to capture the lexical relations more precisely, following Snow et al. [Snow et al., 2004]. This modification cuts short the `conj` edges when nouns are connected by conjunctions such as *and* and *or*. After this shortcut, the syntactic pattern between *authors* and *Herrick* is `ncmod-of:as:dobj-of`, which used to be `ncmod-of:as:dobj-of:and:conj-of`. The syntactic pattern between *Herrick* and *Shakespeare* is `conj-and`, which is a newly introduced special symmetric relation, indicating that the nouns are mutually conjunctional. These conjunction shortcuts are shown in Fig. 7.1 as dotted lines.

Snow et al. also proposed *satellite links*, i.e., single depending nodes attached at the end of paths. We did not include this because they greatly increase the sparseness of the pair-feature relation and it did not help to improve the performance, at least in the preliminary experiment targeting at the current synonym acquisition task.

After extracting syntactic patterns, we replaced some of the words in the patterns by categorical labels to generalize words by categories and to avoid the sparseness of the extracted patterns. This operation includes replacing all occurrences of pronouns with a special label `NP` and cardinal numbers with `MC` and so on. This replacement is conducted based on the CLAWS7 PoS tags of RASP2 outputs. All the replacing operations we employed are listed in Table 7.3.1. For example, if we have `Herrick:dobj:as` as a pattern and the PoS tag of the word `Herrick` is `NP1`, the

Table 7.1: Word generalization operations to syntactic patterns

PoS Tag	Description	Examples	Label after Replacement
MC	cardinal number	*1, 2, 3, ...*	MC
MC1	singular cardinal number	*one, two, ...*	MC
MC2	plural cardinal number	*ones, 60's, ...*	MC
MD	ordinal number	*first, second, next, last, ...*	MD
JB	other adjective*	*360-degree, 64-bit, ...*	*NUM-degree, NUM-bit*
NPD1	singular weekday noun	*Sunday, Monday, ...*	NPD
NPD2	plural weekday noun	*Sundays, Mondays, ...*	NPD
NPM1	singular month noun	*January, February, ...*	NPM
NPM2	plural month noun	*Januaries, Februaries, ...*	(not found)**
NP1	singular proper noun	*Fred, L.A., Claire, ...*	NP
NP2	plural proper noun	*Georges, Palestinians, ...*	NP

* The tag is actually not included in CLAWS7 PoS Tags and can be a RASP2-specific one.
** This PoS tag NPM2 was not found in the corpus we used.

pattern is replaced with NP:dobj:as.

7.3.2 Feature Construction

After the corpus is analyzed and patterns are extracted, the pattern based feature $f_k^P(w_1, w_2)$, which corresponds to the syntactic pattern p_k, is defined as the conditional probability of observing p_k given that the words w_1, w_2 both appear in a sentence. This definition is similar to [Mirkin et al., 2006] and is calculated as:

$$f_k^P(w_1, w_2) = P(p_k|w_1, w_2) = \frac{n'(w_1, w_2, p_k)}{n'(w_1, w_2)}. \tag{7.9}$$

To prevent the frequency counts from varying too much, we took the logarithm of the original count obtained from the corpus, i.e., $n'(w_1, w_2, p_k) = \log(n(w_1, w_2, p_k) + 1)$, where $n(w_1, w_2, p_k)$ is the frequency of pattern p_k co-occurring with words w_1 and w_2, and $n'(w_1, w_2) = \log(n_s(w_1, w_2) + 1) + \sum_{p_k} n'(w_1, w_2, p_k)$, where $n_s(w_1, w_2)$ is the number of times words w_1 and w_2 co-occur in any sentence without having specific relations between them (i.e., no dependency relations with a length of 5 or less are connecting them). Although the formal results are not shown in this chapter, taking logarithm showed about 1% increase on F-1 measure basis on average.

Mirkin et al. also used three additional features besides the one we explained here — aggregated conditional pattern probability, conditional list-pattern probability, and relation direction ratio. As for the conditional list-pattern probability, we did not use this as an additional feature, since the pattern construction method we employed here is dependency relation, which we suppose is powerful enough to capture list structures as well.

Table 7.2: Synonym classifiers to compare

Name	Description	Features Used
DSIM	Distributional Similarity (unsupervised)	f^S
PAT	Pattern-based Features	$f_1^P, ..., f_K^P$
DSIM-PAT	DSIM and Pattern-based Features	$f^S, f_1^P, ..., f_K^P$
CFEAT	Common Features	$f_1^C, ..., f_M^C$
CFEAT-PAT	Common and Pattern-based Features	$f_1^C, ..., f_M^C, f_1^P, ..., f_K^P$
DFEAT	Distributional Features	$f_1^D, ..., f_M^D$
DFEAT-PAT	Distributional and Pattern-based Features	$f_1^D, ..., f_M^D, f_1^P, ..., f_K^P$
ALL	All Features	$f^S, \{f_j^P\}, \{f_i^C\}, \{f_i^D\}$

f^S: distributional similarity, f_j^C: common features, f_j^D: distributional features, f_k^P: pattern-based features. M and K denote the numbers of context types and patterns, respectively.

As for the aggregated conditional pattern probability and relation direction ratio, we included them as additional features and compared the performances but failed to observe any significant improvement in a preliminary experiment. In our case, every relation edge has its counterpart and the value of the relation direction ratio is always 1, because it is the ratio of the probability of a pair (w_1, w_2) having any syntactic relations between them to that of the inversed pair (w_2, w_1). Besides, it is essentially designed for their asymmetric settings of lexical entailment detection task and it therefore cannot be effective for our symmetric setting in the first place.

7.4 Experiments

7.4.1 Synonym Classifiers

Now that we have all features available, eight synonym classifiers are built and compared for the performance. All the classifiers built are listed in Table 7.2, along with their descriptions and the features used. Out of these eight classifiers, the DSIM classifier is the only unsupervised method, where cosine is used to measure the pair similarity. A threshold is set on the similarity and binary classification is performed based on whether the similarity is above or below of the given threshold. How to optimally set this threshold is described later in Section 7.4.2.

The remaining seven classifiers are all SVM-trained, supervised methods. PAT uses only pattern-based features, which is essentially the same as Snow et al.'s method [Snow et al., 2004]. DSIM-PAT is a semi-integrated classifier, which roughly corresponds to Mirkin et al.'s approach [Mirkin et al., 2006]. CFEAT-PAT, DFEAT-PAT, and ALL are fully integrated classifiers which we propose in this chapter.

7.4.2 Experimental Settings

Corpus and Preprocessing

As for the corpus, New York Times section (1994) of English Gigaword[1], consisting of approx. 46,000 documents, 922,000 sentences, and 30 million words, was analyzed by RASP2 to obtain word-context co-occurrences.

This yielded 100,000 or more context types, thus we applied feature selection and reduced the dimensionality. Firstly, we simply applied frequency cut-off to filter out any words and contexts with low frequency, as we have done in Section 3.4.1. In this experiment we set $\theta_w = 5$ and $\theta_c = 5$.

We then applied the type-based context type selection scheme in Section 6.4 to further reduce the dimensionality of vectors. Specifically, we used document frequency (DF) as the importance score and removed the context types with the lowest values of DF until 10% of the original contexts remained. As a result, this process left a total of 3,975 context types, i.e., feature dimensionality.

The feature selection was also applied to pattern-based features to avoid high sparseness — only syntactic patterns which occurred more than or equal to seven times were used. The number of syntactic pattern types left after this process is 17,964.

Supervised Learning

Train and test sets were created as follows: firstly, the nouns listed in the Longman Defining Vocabulary (LDV)[2] were chosen as the target words of classification. Then, all the LDV pairs which co-occur more than or equal to three times with any of the syntactic patterns, i.e., $\{(w_1, w_2) | w_1, w_2 \in \text{LDV}, \sum_p n(w_1, w_2, p) \geq 3\}$ were classified into synonym/non-synonym classes.

To test whether or not a given word pair (w_1, w_2) is a synonym pair, we used the same method as the one we used to evaluate the synonym acquisition task (Section 3.1) — the union of synonyms obtained from the three thesauri (Roget's, COBUILD Thesaurus, and WordNet) was used as the answer set, and the pair (w_1, w_2) is marked as synonyms if and only if w_2 is contained in the answer set of w_1, or w_1 is contained in that of w_2.

All the positive-marked pair, as well as randomly chosen one out of five negative-marked pairs, were collected as the *example set* E. This random selection is to avoid extreme bias toward the negative examples. The example set E ended up with 2,148 positive and 13,855 negative examples, with their ratio being approx. 6.45.

The example set E was then divided into five partitions to conduct five-fold cross validation, of which four partitions were used for learning and the one for testing. SVM^{light} [Joachims, 1999] was adopted for machine learning. We used the linear and RBF (radial basis functions) kernels. The parameters, i.e., the similarity threshold of DSIM classifier, gamma parameter γ of the RBF kernel, and the cost-factor j of

[1] http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2003T05

[2] http://www.cs.utexas.edu/users/kbarker/working_notes/ldoce-vocab.html

Table 7.3: Performance comparison of synonym classifiers

Classifier	Linear kernel			RBF kernel		
	Precision	Recall	F-1	Precision	Recall	F-1
DSIM	35.56%	50.18%	41.62%	—	—	—
PAT	23.59%	41.10%	29.97%	23.82%	47.03%	31.62%
DSIM-PAT	31.83%	52.27%	39.57%	30.46%	61.30%	40.69%
CFEAT	55.29%	82.13%	66.09%	89.30%	**83.18%**	86.13%
CFEAT-PAT	**55.41%**	82.02%	66.14%	89.31%	**83.18%**	86.13%
DFEAT	53.71%	**87.82%**	**66.65%**	**95.37%**	82.31%	88.36%
DFEAT-PAT	53.67%	**87.60%**	66.56%	95.26%	82.42%	**88.38%**
ALL	**59.31%**	85.45%	**70.02%**	**99.64%**	80.50%	**89.05%**

SVM, i.e., the ratio by which training errors on positive examples outweigh errors on negative ones, were optimized using one of the five-fold cross validation train-test pair as a development set on the basis of F-1 measure. The optimal values were grid-searched on $j = 6, 7, 8, 9, 10$ for the linear kernel, and $j = 6, 7, 8, 9, 10$ and $\gamma = 0.001, 0.005, 0.01, 0.05, 0.1$ for the RBF kernel. The performance was evaluated for the other four train-test pairs and the average values were recorded.

7.4.3 Performance Comparison

The performances, i.e., precision, recall, and F-1 measure, of the seven classifiers were evaluated and shown in Table 7.3. DSIM classifier is unsupervised, while the other six classifiers are trained using both the linear and RBF kernels.

Firstly, when we compare DSIM and DSIM-PAT, the latter showed slight increase on recall, but not on precision. F-1 measure did not improve even when the RBF kernel is used. This means that, even though integrating pattern-based features did increase the recall, we could not confirm the positive result of Mirkin et al.'s integrated approach [Mirkin et al., 2006] for this synonym acquisition task.

On the other hand, when common or distributional features are used instead of distributional similarity, we observed drastic performance improvement — both precision and recall improved and F-1 measure increased by more than 50%, comparing DSIM and CFEAT/DFEAT. The improvement is greater for DFEAT than for CFEAT, especially when the RBF kernel is used.

Further adding PAT features (CFEAT-PAT and DFEAT-PAT) to common/ distributional features increased the performance slightly for CFEAT, but not for DFEAT at all. This implies that the discriminative ability of distributional features were so high that pattern-based features were almost redundant.

The reason of the drastic improvement of CFEAT/DFEAT is that, as far as we speculate, the supervised learning may have favorably worked to cause the same effect as automatic feature selection technique. Features with high discriminative power may have been automatically promoted. In the distributional similarity setting

and Mirkin et al.'s integrated approach, in contrast, the contributions of context types are uniformly fixed. Furthermore, using the RBF kernel (as well as some other kernels such as polynomial kernels) corresponds to mapping the input vectors to a higher-dimensional space, which includes higher degrees of combinations of input features. Through this process, the classifiers might have been able to find useful feature combinations for discrimination. This led to the superior performance of the RBF kernel compared to the liner one.

To note, the performance of PAT was the lowest, reflecting the fact that synonym pairs rarely occur in the same sentence, making the identification using only syntactic pattern clues even more difficult. On the contrary, Mirkin et al.'s task is the automatic acquisition of lexical entailment relations, which are essentially asymmetric. It is for the same reason that adding pattern-based features to other features showed little contribution to classification performance except for CFEAT.

We further compared the performance of the all-in-one classifier ALL with that of other classifiers. It greatly increased precision compared to all the other methods, even approx. 5% higher than CFEAT/DFEAT. On the other hand, although it did not achieve as high recall as CFEAT/DFEAT did, the overall F-1 measure improved to 4% higher than DFEAT with the linear kernel and 1% higher with the RBF kernel[3]. Although ALL achieved one of the highest precision/F-1 levels, notice that it requires more than twice the size of the feature space compared to the others. It is also computationally extensive. Therefore, the trade-off between computational cost and performance gain should be taken into consideration in practice.

7.4.4 Extracted Synonyms

In the second part of the experiment, we further investigated what kind of synonyms were actually acquired by the classifiers we built. This is an open test which targets at the extraction of non-LDV words. The reason why we conducted this is that we cannot rule out the possibility that the high performance of the DFEAT-based methods seen in the previous experiment was simply due to the rather limited target word settings.

The rest of the experimental setting was almost the same as the previous one — we used the same model trained on the example set E. The RBF kernel was again used for SVM-based classifiers. The only difference is that the test set also included non-LDV words, i.e., all the words which appeared more than θ_w times in the corpus. Because including PAT features did not yield a significant difference in the previous experiments except for CFEAT, we only paid attention to the DSIM, CFEAT, and DFEAT classifiers. Here the task is actually to *rank* synonyms, not to classify them. This is done by firstly fixing one word, which we call *query word*, and then computing similarities of all the other words in the vocabulary and ranking them by the similarity in a descending order. The SVM-based classifiers cannot compute

[3]We also tried the CFEAT-DFEAT classifier, i.e., ALL without PAT and DSIM. It did not show any significant difference compared to ALL, which means that the combination CFEAT-DFEAT is so dominant that PAT and DSIM were simply redundant in this case.

Table 7.4: Acquired synonyms of *opera*

Rank	DSIM	CFEAT	DFEAT
1	movie	imitator	movie
2	dance	dance	concert
3	music	sitcom	comedy
4	ballet	ballet	music
5	art	oratorio	ballet
6	concert	movie	film
7	song	concert	song
8	film	music	dance
9	television	repertory	march
10	theater	jazz	show

Table 7.5: Acquired synonyms of *continent*

Rank	DSIM	CFEAT	DFEAT
1	washcloth	region	region
2	country	washcloth	country
3	mudstone	ration	nation
4	crackhead	cameraman	ensemble
5	veldt	country	land
6	hearth	coil	ocean
7	kolopaking	congregant	shelf
8	epidermis	rafter	avenue
9	taiga	childhood	precinct
10	region	shack	plane

Table 7.6: Acquired synonyms of *purse*

Rank	DSIM	CFEAT	DFEAT
1	instill	camisole	dog
2	camisole	lode	rifle
3	saving	gazar	jewel
4	tangerine	liabilty	something
5	circulation	pantsuit	hat
6	prize	instill	reader
7	liability	tunic	hundred
8	demask	vein	photo
9	atonement	scotch	bag
10	reward	kimono	blend

the similarity directly, so we used the value of decision function of SVM, i.e., the distance from the maximum-margin hyperplane.

We chose the three query words in this section from non-LDV words, which guarantees that this is an open test (remember that the example set E consists only of LDV words). The first case is the top 10 synonyms acquired for the word *opera*, which are shown in Table 7.4. This is a case where the outputs of DSIM, CFEAT, and DFEAT are almost as good as others. In addition to the three query words listed in this section, we manually checked 15 query words, and found out that this was the most common case. Here even DSIM performs quite well, and so do the others. Although the result of CFEAT seems to include the fewer number of relevant words, it does include "interesting" related words instead, such as *oratorio* and *repertory*. This may explain the slightly lower precision and higher recall of CFEAT compared with DFEAT.

Table 7.5 shows the synonyms for the word *continent*. This is another case, where DSIM performed poorly. CFEAT also failed to identify appropriate synonyms, although its ranking is slightly different from DSIM's. On the other hand, DFEAT

successfully listed many related words to *continent*. Along with the previous experiment, this result suggests the superiority of DFEAT compared to DSIM.

Finally, Table 7.6 shows the synonyms for the word *purse*. This is yet another case, where all the three classifiers have difficulty finding correct synonyms. The result of DSIM is apparently filled with a number of unrelated words, perhaps except for the word *camisole*. We suspect that this is because the word *purse* is unlikely to appear in most news articles. Even when it does, the use of the word can be different from the normal usage. Still, CFEAT began listing some related words such as *pantsuit*, *tunic*, and *kimono*, although the number is not large. DFEAT also managed to extract something you wear (e.g. *jewel*, *hat*, and *bag*) and this is definitely a step in a right direction. This result also shows the robustness of the classification-based approaches to synonym acquisition.

7.4.5 Error Anaylysis

To further investigate the cause of errors the classifiers made, we looked into the misclassified examples in the test set. Because we used DFEAT's best result in the following analysis, the examples below can be the ones which all of the classifiers find difficulty in classifying.

Firstly, we pay attention to false negatives, i.e., examples for which the annotated answer is positive (synonym) but the output of the classifier was not (non-synonym). The pair (*word*, *news*) is one of them. As we can easily guess, the reason of misclassification is that these are synonyms only in limited cases (e.g. *spread the word*). In effect, we notice that these two words do not share significant amount of context, except for `(ncsubj leak * _)` (both words can be the subject of the verb *leak*), `(ncsubj spread * obj)` (both words can be the subject of the verb *spread* in a passive sentence), and so on. We found some other pairs, such as (*damage*, *cost*) and (*level*, *standard*), which fall into this category.

We also found another category of false negatives, which we call *out-of-domain words*. This one includes pairs such as (*soul*, *body*) and (*path*, *road*). These pairs may have been misclassified because they consist of words that are general enough but not exactly related to news articles, and a sufficient amount of context was not obtained from the corpus we used. Adding general-domain corpora could solve this problem.

Secondly, we looked into false positives, i.e., examples for which the annotated answer is negative (non-synonym) but the output of the classifier was not (synonym). Surprisingly, we found out that many of false positives were actually *related* in some sense. For example, it included pairs such as (*stage*, *part*), (*eye*, *mind*), and (*corner*, *end*), which in fact share a lot of context types — `(det * latter)`, `(ncmod _ * early)`, `(ncmod _ * crucial)`, etc. for (*stage*, *part*), `(ncmod poss * people)`, `(ncmod poss * everyone)`, `(ncmod _ * suspicious)`, etc. for (*eye*, *mind*), and `(ncmod _ * opposite)`, `(ncmod _ * upper)`, `(dobj near *)`, etc. for (*corner*, *end*). Although these are not exact synonyms, the classifier didn't completely fail to capture related words, which could be candidates for synonyms.

The negative examples are "contaminated" with such related words because we created them by ensuring that they are not included in the thesauri we used, although

this does not necessarily ensure that they are not even *related*. Further refining the train set by manually ensuring that they don't include any related pairs can eliminate such false positives. In general, the performance and the behavior of our classifiers could be greatly affected by the train sets used — for example, we expect that it is possible to extract exact synonyms by constraining the train set to include only the "tightly" related words. This also might enable to exclude antonyms and cohyponyms from extracted related terms, as Lin et al. [Lin et al., 2003] did. The effect of replacing the train sets should be investigated in the future.

7.4.6 Effective Features

We are now naturally interested in which common/distributional features, i.e., which of $f_1^C, ..., f_M^C$ or $f_1^D, ..., f_M^D$, were the most useful for the task. Although there can be a number of methods to rank or select features for machine learning, here we simply evaluated the effectiveness of a feature (or possibly a set of features) by removing the target features and re-evaluate the overall performance of the classifier. The bigger the difference of the performance is, the more likely the removed features are to be important.

However, it is impractical to investigate the individual contribution of every single distributional/common feature in this way because there are so many (3,975 in number) of them. Therefore we firstly investigate the contribution of *feature categories*, i.e., sets of features grouped by their syntactic relations. In this experiment, we evaluated the difference of performance (F-1) compared to the classifier performance with all the features, by taking out one category at a time. We used the DSIM classifier and the linear kernel because the RBF kernel considers the higher degree of correlation between features as we mentioned in Section 6.3 and this might make it difficult to evaluate the contributions of features individually.

Figure 7.2 shows the result of this experiment. The amount of F-1 decrease of the feature category is shown, along with the number of the context types that the category contains. Context categories with 10 or fewer context types are omitted here. It clearly shows that the categories `dobj`, `ncmod`, and `ncsubj`[4] demonstrate the best performance. These are also the categories which include the highest numbers of context types. This suggests that categories with more context types tend to achieve higher performance. Indeed, there is a high correlation ($\rho = 0.95$) between the performance decrease and the number of context types. It is not the quality but the size of the categories that largely determines its contribution.

We then randomly extracted five context types as samples from `dobj`, `ncmod`, and `ncsubj` and evaluated their contributions in the same way as we investigated the contribution of context categories. The result is shown in Table 7.7, where the amount of F-1 decrease and the frequency of the context type in the example set are shown. Although the two highest values in each category, which are emphasized in bold face, suggest that high-frequency features tend to contribute to the performance

[4] `dobj`, `ncmod`, and `ncsubj` stand for direct object, non-clausal modification, and non-clausal subject, respectively.

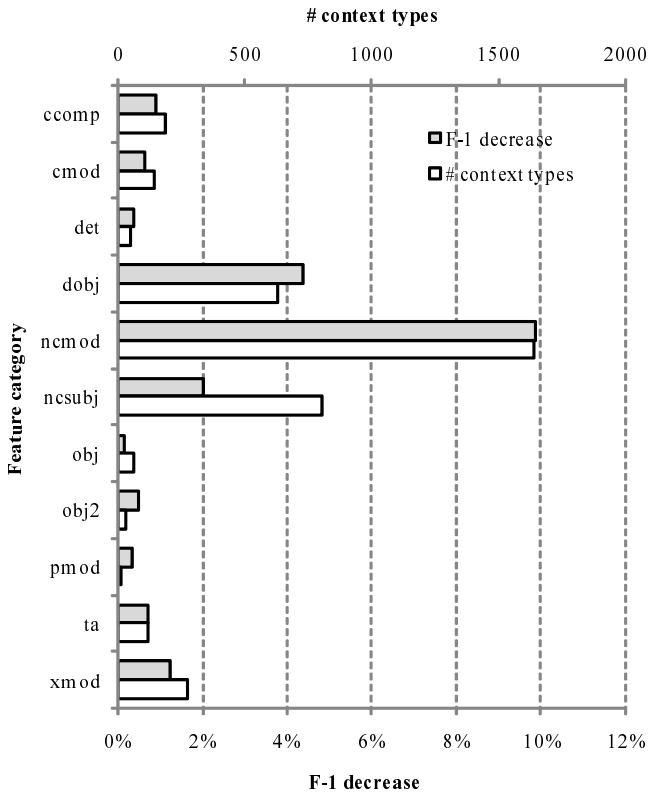

Figure 7.2: Feature categories and their contributions

Table 7.7: Performance contribution of individual features

Feature	F-1 decrease	Frequency
dobj:need:*	**0.26%**	**1650**
dobj:explore:*	0.12%	284
dobj:produce:*	**0.33%**	**1015**
dobj:upon:*	0.18%	468
dobj:cite:*	0.23%	578
ncmod:_:*:previous	**0.35%**	924
ncmod:_:*:right	0.23%	**1986**
ncmod:_:*:city	0.18%	**976**
ncmod:_:*:road	**0.29%**	396
ncmod:_:two:*	0.09%	93
ncsubj:earn:*:_	0.07%	296
ncsubj:lead:*:_	**0.17%**	**1525**
ncsubj:carry:*:_	0.15%	643
ncsubj:make:*:_	**0.30%**	**6160**
ncsubj:convert:*:_	0.15%	69

more, the correlation is less evident ($\rho = 0.43$) than that of context categories and some of the features, e.g., `ncmod:_:right` and `ncmod:_:city` do not perform well in spite of their high frequency. The result implies that the specificity of context types correlates with their performance, because `ncmod:_:*:right` has quite a broad meaning, while `ncmod:_:*:road` does not. Further investigation on feature types and their performance contribution is the future work.

7.5 Learning Similarity Measure

In this section we pursue a different approach to supervised synonym acquisition, where the similarity/distance measure, which we call *metric* in this section, is trained using some pre-compiled lexical resources such as WordNet. In other words, our proposal here is to extend and fine-tune the corpus-based approach with the training data obtained from the resource-based approach. We apply metric learning to this task. Although still in their infancy, distance metric learning methods have undergone rapid development in the field of machine learning. In a setting similar to semi-supervised clustering, where known instances of similar or dissimilar objects are given, a metric such as the Mahalanobis distance can be learned from a few data points and tailored to fit a particular purpose. Although classification methods such as logistic regression now play important roles in natural language processing, the use of metric learning has yet to be explored.

Since popular current methods for synonym acquisition require no statistical learning, it seems that supervised machine learning should easily outperform them. Unfortunately, there are obstacles to overcome. Since metric learning algorithms usu-

ally learn the parameters of a Mahalanobis distance, the number of parameters is quadratic to the number of features. They learn how two features should interact to produce the final metric. While traditional metrics forgo examining of the interactions entirely, in applying metrics such as Jaccard coefficient, it is not uncommon nowadays to use more than 10,000 features, a number that a typical metric learner is incapable of processing. Thus we have two options: one is to find the most important features and model the interactions between them, and the other is simply to use a large number of features. We experimentally examined the two options and found that metric learning is useful in synonym acquisition, despite it utilizing fewer features than traditional methods.

7.5.1 Metric Learning

Problem Formulation

To set the context for metric learning, we first describe the objects whose distances from one another we would like to know. Our object is the context vector of a target word, as we have shown in the previous chapters. The vectors of two target words represent their contexts as points in a multidimensional feature-space R^M. A suitable metric (for example, Euclidean) defines the distance between the two points, thereby estimating the semantic distance between the target words.

Given points $x_i, x_j \in R^M$, the (squared) Mahalanobis distance between them is parameterized by a positive definite matrix A as:

$$d_A(x_i, x_j) = (x_i - x_j)^T A(x_i - x_j). \tag{7.10}$$

The Mahalanobis distance is a straightforward extension of the standard Euclidean distance. If we let A be the identity matrix, the Mahalanobis distance reduces to the Euclidean distance. Our objective is to obtain the positive definite matrix A that parameterizes the Mahalanobis distance between the vectors of two synonymous words is small, while the distance between the vectors of two dissimilar words is large. Stated more formally, the Mahalanobis distance between two similar points must be smaller than a given upper bound, i.e., $d_A(x_i, x_j) \leq u$ for a relatively small value of u. Similarly, two points are dissimilar if $d_A(x_i, x_j) \geq l$ for sufficiently large l.

As we discussed in Chapter 3, we were able to use the Euclidean distance to acquire synonyms quite well, as long as the vectors are L2-normalized. Therefore, we would like the positive definite matrix A of the Mahalanobis distance to be close to the identity matrix I. This keeps the Mahalanobis distance similar to the Euclidean distance, which would help to prevent over-fitting the data. To optimize the matrix, we follow the information theoretic metric learning approach described in [Davis et al., 2007]. We summarize the problem formulation advocated by this approach in this section and the learning algorithm in the next section.

To define the closeness between A and I we use a simple bijection (up to a scaling function) from the set of Mahalanobis distances to the set of equal mean multivariate Gaussian distributions. Without loss of generalization, let the equal mean be μ. Then given a Mahalanobis distance parameterized by A, the corresponding Gaussian

is $p(x; A) = \frac{1}{Z}\exp(-\frac{1}{2}d_A(x, \mu))$ where Z is the normalizing factor. This enables us to measure the distance between two Mahalanobis distances with the Kullbuck-Leibler (KL) divergence of two Gaussians:

$$KL(p(x; I)||p(x; A)) = \int p(x, I) \log\left(\frac{p(x; I)}{p(x; A)}\right) dx. \tag{7.11}$$

Given pairs of similar points S and pairs of dissimilar points D, the optimization problem is:

$$\begin{array}{ll} \min_A & KL(p(x; I)||p(x; A)) \\ \text{subject to} & \begin{array}{ll} d_A(x_i, x_j) \leq u & (i, j) \in S \\ d_A(x_i, x_j) \geq l & (i, j) \in D \end{array} \end{array} \tag{7.12}$$

Learning Algorithm

[Davis and Dhillon, 2006] has shown that the KL divergence between two multivariate Gaussians can be expressed as the convex combination of a Mahalanobis distance between mean vectors and the LogDet divergence between covariance matrices. The LogDet divergence equals

$$D_{ld}(A, A_0) = tr(AA_0^{-1}) - \log\det(AA_0^{-1}) - N \tag{7.13}$$

for N by N matrices A and A_0. If we assume the means of the Gaussians to be the same, we have

$$KL(p(x; A_0)||p(x, A)) = \frac{1}{2}D_{ld}(A, A_0) \tag{7.14}$$

The optimization problem can be restated as

$$\begin{array}{ll} \min_{A\succeq 0} & D_{ld}(A, I) \\ \text{subject to} & \begin{array}{ll} tr(A(x_i - x_j)(x_i - x_j)^T) \leq u & (i, j) \in S \\ tr(A(x_i - x_j)(x_i - x_j)^T) \geq l & (i, j) \in D \end{array} \end{array} \tag{7.15}$$

We then incorporate slack variables into the formulation to guarantee the existence of a feasible solution for A. The optimization problem becomes:

$$\begin{array}{ll} \min_{A\succeq 0} & D_{ld}(A, I) + \gamma D_{ld}(diag(\xi), diag(\xi_o)) \\ \text{subject to} & \begin{array}{ll} tr(A(x_i - x_j)(x_i - x_j)^T) \leq \xi_{c(i,j)} & (i, j) \in S \\ tr(A(x_i - x_j)(x_i - x_j)^T) \geq \xi_{c(i,j)} & (i, j) \in D \end{array} \end{array} \tag{7.16}$$

where $c(i, j)$ is the index of the (i, j)-th constraint and ξ is a vector of slack variables whose components are initialized to u for similarity constraints and l for dissimilarity constraints. The tradeoff between satisfying the constraints and minimizing $D_{ld}(A, I)$ is controlled by the parameter γ. To solve this optimization problem, the algorithm shown in Figure 7.3 repeatedly projects the current solution onto a single constraint.

Input:
X (N by M matrix), I (identity matrix)
S (set of similar pairs), D (set of dissimilar pairs)
γ (slack parameter), c (constraint index function)
u, l (distance thresholds)
Output:
A (Mahalanobis matrix)

$A := I$
$\lambda_{ij} := 0$
$\xi_{c(ij)} := u$ for $(i,j) \in S$; otherwise, $\xi_{c(i,j)} := l$
repeat
  Pick a constraint $(i,j) \in S$ or $(i,j) \in D$
  $p := (x_i - x_j)^T A (x_i - x_j)$
  $\delta := 1$ if $(i,j) \in S$, -1 otherwise.
  $\alpha := \min(\lambda_{ij}, \frac{\delta}{2}(\frac{1}{p} - \frac{\gamma}{\xi_{c(i,j)}}))$
  $\beta := \delta\alpha / (1 - \delta\alpha\xi_{c(i,j)})$
  $\xi_{c(i,j)} := \gamma\xi_{c(i,j)} / (\gamma + \delta\alpha\xi_{c(i,j)})$
  $\lambda_{ij} := \lambda_{ij} - \alpha$
  $A := A + \beta A(x_i - x_j)(x_i - x_j)^T A$
until convergence
return (A)

Figure 7.3: Information Theoretic Metric Learning Algorithm

7.5.2 Experimental Settings

In this section, we describe the experimental settings including the preprocessing of data and features, creation of the query word sets, and settings of the cross validation.

We used dependency-based context (`dbc`) as the context for words, as we did in Chapter 3. The corpus was July 1994 portion of New York Times, which consists of 7,593 documents and approx. 5 million words. PMI was used as the weighting function. In order to reduce the number of features, the cut-off frequency with $\theta_w = \theta_c = 5$ and the DF (90% reduction) feature selection method were used. These feature reduction operations reduced the dimensionality to a figure as small as 1,281, while keeping the performance loss at minimum.

To formalize the experiments, we must prepare a set of query words for which synonyms are known in advance. Again, we used the 771 LDV words (Section 3.1.1), from which we extracted 231 words that had five or more synonyms in the combined thesaurus. We selected these 231 words to be the query words and distributed them into five partitions so as to conduct five-fold cross validation.

Four partitions were used in training, and the remaining partition was used in testing. For each fold, we created the training set from four partitions as follows; for each query word in the partitions, we randomly selected five synonymous words and added the pairs of query words and synonymous words to S, the set of similar pairs. Similarly, five pairs of query words and dissimilar words were randomly added to D, the set of dissimilar pairs. The training set for each fold consisted of S and D. Since a learner trained on an imbalanced dataset may not learn to discriminate enough between classes, we sampled dissimilar pairs to create an evenly distributed training dataset.

To make the evaluation realistic, we used a different method to create the test set: we paired each query word with each of the 771 remaining words to form the test set. Thus, in each fold, the training set had an equal number of positive and negative pairs, while in the test set, negative pairs outnumbered the positive pairs. While this is not a typical setting for cross validation, it renders the evaluation more realistic since an automatic synonym acquisition system in operation must be able to pick a few synonyms from a large number of dissimilar words.

The meta-parameters of the metric learning model were simply set $u = 1$, $l = 2$ and $\mu = 1$. Each training set consisted of 1,850 pairs, and the test set consisted of 34,684 pairs. Since we conducted five-fold cross validation, the reported performance in this experiment is actually a summary over different folds.

7.5.3 Evaluation Measures

In this experiment, we used an evaluation program for KDD Cup 2004 [Caruana et al., 2004] called Perf to measure the effectiveness of the metrics in acquiring synonyms. To use the program, we used the following formula to convert each distance metric to a similarity metric: $s(x_i, x_j) = 1/(1 + \exp(d(x_i, x_j)))$. Below, we summarize the three measures we used: Mean Average Precision, TOP1, and Average Rank of Last Synonym.

Mean Average Precision (APR) Perf implements a definition of average precision sometimes called "expected precision." Perf calculates the precision at every recall where it is defined. For each of these recall values, Perf finds the threshold that produces the maximum precision, and takes the average over all of the recall values greater than 0. Average precision is measured on each query, and then the mean of each query's average precision is used as the final metric. A mean average precision of 1.0 indicates perfect prediction. The lowest possible mean average precision is 0.0.

Average Rank of Last Synonym (RKL) As in other evaluation measures, synonym candidates are sorted by predicted similarity, and this metric measures how far down the sorted cases we must go to find the last true synonym. A rank of 1 indicates that the last synonym is placed in the top position. Given a query word, the highest obtainable rank is k if there are k synonyms in the corpus. The lower this measure is the better. Average ranks near 771 indicate poor performance.

TOP1 In each query, synonym candidates are sorted by predicted similarity. If the word that ranks at the top (highest similarity to the query word) is a true synonym of the query word, Perf scores a 1 for that query, and 0 otherwise. If there are ties, Perf scores 0 unless all of the tied cases are synonyms. TOP1 score ranges from 1.0 to 0.0. To achieve 1.0, perfect TOP1 prediction, a similarity metric must place a true synonym at the top of the sorted list in every query. In the next section, we report the mean of each query's TOP1.

7.5.4 Results

The evaluations of the metrics are listed in Table 7.8. The figure on the left side of $\rightarrow$ represents the performance with 1,281 features, and that on the right side with 12,812 features. The metric names with "L2" at the end show that L2-normalized vectors were used. Of all the metrics in Table 7.8, only the Mahalanobis L2 is trained with the previously presented metric learning algorithm. Thus, the values for the Mahalanobis L2 are produced by the five-fold cross validation, while the rest are given by the straight application of the metrics discussed in Section 4.2 to the same dataset. Strictly speaking, this is not a fair comparison, since we ought to compare a supervised learning with a supervised learning. However, our baseline is not the simple Euclidean distance; it is the Jaccard coefficient and cosine similarity, a handcrafted, best performing metric for synonym acquisition, with 10 times as many features.

The computational resources required to obtain the Mahalanobis L2 results were as follows: in the training phase, each fold of cross validation took about 80 iterations (less than one week) to converge on a Xeon 5160 3.0GHz. The time required to use the learned distance was a few hours at most.

At first, we were unable to perform competitively with the Euclidean distance. As seen in Table 7.8, the TOP1 measure of the Euclidean distance is only 1.732%. This indicates that the likelihood of finding the first item on the ranked list to be a true synonym is 1.732%. The vector-based Jaccard coefficient performs much better than

Table 7.8: Evaluation of Various Metrics, as Number of Features Increases from 1,281 to 12,812

Metric	APR	RKL	TOP1
Cosine	0.1184 → 0.1324	580.27 → 579.00	0.2987 → 0.3160
Euclidean	0.0229 → 0.0173	662.74 → 695.71	0.0173 → 0.0000
Euclidean L2	0.1182 → 0.1324	580.30 → 578.99	0.2943 → 0.3160
Jaccard-s	0.1120 → 0.1264	580.76 → 579.51	0.2684 → 0.2943
Jaccard-s L2	0.1113 → 0.1324	580.29 → 570.88	0.2640 → 0.2987
Jaccard-v	0.1189 → 0.1318	580.50 → 580.19	0.3073 → 0.3030
Jaccard-v L2	0.1184 → 0.1254	580.27 → 570.00	0.2987 → 0.3160
JS	0.0199 → 0.0170	681.97 → 700.53	0.0129 → 0.0000
JS L2	0.0229 → 0.0173	679.21 → 699.00	0.0303 → 0.0086
Manhattan	0.0181 → 0.0168	687.73 → 701.47	0.0043 → 0.0000
Manhattan L2	0.0185 → 0.0170	686.56 → 701.11	0.0043 → 0.0086
SD99	0.0324 → 0.1039	640.71 → 588.16	0.0173 → 0.2640
SD99 L2	0.0334 → 0.1117	633.32 → 586.78	0.0216 → 0.2900
Mahalanobis L2	**0.1866**	**545.09**	**0.4545**

the Euclidean distance, placing a true synonym at the top of the list 30.736% of the time.

Table 7.9 shows the Top 10 Words for Query "branch". The results for the Euclidean distance rank "hut" and other dissimilar words highly. This is because the norm of such vectors is small, and in a high dimensional space, the sparse vectors near the origin are relatively close to many other sparse vectors. To overcome this problem, we normalized the input vectors by the L2 norm $x' = x/||x||$. This normalization enables the Euclidean distance to perform very much like the cosine similarity, since the Euclidean distance between points on a sphere acts like the angle between the vectors. Surprisingly, normalization by L2 did not affect other metrics all that much; while the performances of some metrics improved slightly, the L2 normalization lowered that of the Jaccardv metric.

Once we learned the normalization trick, the learned Mahalanobis distance consistently outperformed all other metrics, including the ones with 10 times more features, in all three evaluation measures, achieving an APR of 18.66%, RKL of 545.09 and TOP1 of 45.455%.

7.5.5 Discussion

Examining the learned Mahalanobis matrix revealed interesting features. The matrix essentially shows the covariance between features. While it was not as heavily weighted as the diagonal elements, we found that its positive non-diagonal elements were quite interesting. They indicate that some of the useful features for finding synonyms are correlated and somewhat interchangeable. The example includes a pair of features, `(dobj begin *)` and `(dobj end *)`. It was a pleasant surprise to see

Table 7.9: Top 10 Words for Query "branch"

	Cosine	Euclidean	Euclidean L2	Jaccard	Jaccardv	Mahalanobis L2
1	*office	hut	*office	*office	*office	*division
2	area	wild	area	border	area	group
3	*division	polish	*division	area	*division	*office
4	border	thirst	border	plant	border	line
5	group	hollow	group	*division	group	period
6	organization	shout	organization	mouth	organization	organization
7	store	fold	store	store	store	*department
8	mouth	dear	mouth	circle	mouth	charge
9	plant	hate	plant	stop	plant	world
10	home	wake	home	track	home	body

that one implies the other. Among the diagonal elements of the matrix, one of the heaviest features was being the direct object of "by." This indicates that being the object of the preposition "by" is a good indicator that two words are similar. A closer inspection of the NYT corpus showed that this preposition overwhelmingly takes a person or organization as its object, indicating that words with this feature belong to the same class of a person or organization. Similarly, the class of words that "to" and "within," take as an objects were clear from the corpus: "to" takes a person or place, "within" takes duration of time[5]. Other heavy features includes being the object of "write" or "about." While not obvious, we postulate that having these words as a part of the context indicates that a word is an event of some type.

7.6 Conclusion

In this chapter, we proposed a novel approach to automatic synonym identification based on supervised learning technique and new context-based features. We firstly re-formalized the synonym acquisition task as a classification problem, and then proposed context-based features called *distributional features*, as opposed to *common features* which have been used in the conventional methods. This formalization enabled to use both context-based features and pattern-based features and to build fully integrated classifiers.

In the experiments, we built eight classifiers based on distributional, common, and/or pattern-based features. The experimental results showed that context-based features showed a great increase (more than 60% on F-1 measure) compared to distributional similarity-based methods. On the other hand, pattern-based features were only partially effective when combined with common features whereas with distributional features they were simply redundant. Including all the features did increase

[5]Interestingly, we note that not all prepositions were as heavy: "beyond" and "without" were relatively light among the diagonal elements. In the NYT corpus, the class of words they take was not as clear as, for example, "by."

precision and F-1 measure, at the expense of a slight reduction of recall and high computational complexity.

In the latter part of this chapter, we applied metric learning to automatic synonym acquisition for the first time in the field. Our experiments showed that the learned metric significantly outperforms existing similarity metrics. This outcome indicates that while we must resort to feature selection to apply metric learning, the performance gain from the supervised learning is enough to offset the disadvantage and justify its usage in some applications. This leads us to think that a combination of the learned metric with unsupervised metrics with even more features may produces the best results. We also discussed interesting features found in the learned Mahalanobis matrix. Since metric learning is known to boost clustering performance in a semi-supervised clustering setting, we believe these automatically identified features would be helpful in assigning a target word to a word class.

The impact of this study is that it makes unnecessary to carefully choose similarity measures such as cosine and Jaccard's. Instead, features, or similar/dissimilar pairs, can be directly given to supervised learning methods right after their construction. There are still some issues to address as the current approach is only in its infancy. For example, the formalization of distributional features requires further investigation. Although we adopted total correlation this time, there could be some other construction methods which may show higher performance.

Another contribution of this study is that it introduced supervised learning approaches to synonym acquisition. This makes it possible to use various techniques proposed in the machine learning field. For example, the use of sophisticated kernels such as feature conjunction [Oyama and Manning, 2004] is worth careful consideration. How they affect the synonym acquisition performance should be investigated in the future.

Chapter 8

Conclusion

8.1 Conclusion of This Thesis

In this thesis, we addressed the formalization, extension, selection, and modeling problems of context in order to achieve better synonym acquisition from large corpora, which were all unsolved or underestimated issues in the literature of synonym acquisition and distributional similarity.

In Chapter 2, we firstly introduced the basic concept of distributional similarity. We defined the context of a word as the set of pieces of lexical information associated with the word, and the distributional similarity as the similarity between two context sets. We then described how to extract dependency-based context, and introduced a number of similarity measures and weighting functions, as the baseline for other experiments described in this thesis.

In the succeeding Chapter 3, we actually applied the concepts introduced in Chapter 2 to the synonym acquisition task. To evaluate the performance, we defined two evaluation measures, i.e., average precision (AP) and correlation coefficient (CC), which both use existing thesauri such as WordNet. As the preprocessing of experiments, we firstly investigated the effect of frequency cut-off to word and context types, which has been used in all the experiments described in this thesis. We then compared similarity measures and weighting functions introduced in Chapter 3, and showed that cosine similarity (`cosine`), vector-based Jaccard coefficient (`jaccard-v`), and Jensen-Shannon divergence (`js`) performed well, meaning that some type of normalization to the context vector or the probability distribution is indispensable. The experiment also showed that PMI and t-test were among the best, meaning that the word-based normalization factor they have may have helped.

In Chapter 4, we introduced and applied two latent semantic models, i.e., LSI and PLSI, to synonym acquisition and compared their performance with the simple VSM. Although each model behave quite differently, LSI and PLSI achieved higher AP/CC values when coupled with their own suitable measures, which were `jaccard-v` for LSI and `js` for PLSI. We also showed that the performance peaked around $K = 100$ to 150.

In Chapter 5, we formalized and extended the normal direct dependency to cover

indirectly related words and enhance the contextual information for distributional similarity. The experiment showed that incorporating indirect dependency in addition to direct dependency was effective for the acquisition performance. The improvement was especially clear when fine-grained context representations, namely, word path (`WP`) and full path (`FP`) were used.

In Chapter 6, we proposed three schemes of context selection for distributional similarity: category-, type-, and co-occurrence based selection. In the experiment, these three selection schemes were compared and it clarified the characteristics of the schemes and the measures. It showed the effectiveness of the simplest category-based selection, while type- and co-occurrence based selection methods worked well for both `wbc` and `dbc`, showing that these method can be generally and flexibly used for any kinds of context and dimensionality/computational cost constraints.

In Chapter 7, two novel, supervised approaches to synonym acquisition were proposed. The first one is a classification model, where we proposed context-based features called distributional features, which enabled to use both context-based features and pattern-based features to build fully integrated classifiers. The comparison experiment showed that the context-based features showed a great increase (more than 60% on F-1 measure) compared to distributional similarity-based methods. The other approach learned Mahalanobis distance, which significantly outperformed existing similarity metrics. Although we had to resort to aggressive feature reduction to make it possible to apply the learning, the performance gain from the supervised learning is enough to offset the disadvantage and justify its usage in some applications.

8.2 Future Direction

As for the topics we described in Chapter 2, there can still be more similarity measures and weighting functions which are effective for synonym acquisition. One way to extend the existing weighting measures is to consider the weighted sum of them, as Weed and Weir proposed [Weeds and Weir, 2003]. Another approach is to *learn* them from some training data, as we have shown in Chapter 7. Effective measures and weights can be different from application to application. Their behavior should be further investigated in the future.

Related to the latent semantic models we used in Chapter 4, there is some recent development in the field such as Latent Dirichlet Allocation (LDA) [Blei et al., 2003]. The common criticism to PLSI is that it is not a formal generative model for document (it only models the generation of document and context types from latent classes) Thus we have to see the power of formally-defined generative models in synonym acquisition. Paying attention to the dimensionality reduction, there are still a number of methods related to this issue, including Random projection [Bingham and Mannila, 2001] and Locality Sensitive Hashing (LSH) [Ravichandran et al., 2005]. Comprehensive comparison of these methods in synonym acquisition is future work.

As for the context formalization we discussed in Chapter 5, we still have some problems of context weighting and correlation. Appropriately weighting context types becomes more and more difficult as the context gets more complex, like indirect

dependency we proposed. Pado and Lapata [Pado and Lapata, 2007] tackled this issue by introducing the *path value function*, while the formalization of weighting functions which incorporates the context representations is necessary.

In Chapter 6, we only presented a considerably simple classification model of distributional similarity to assign importance scores for distributional similarity. Further refined models could be considered, and their effectiveness for type-based context selection should be investigated in the future. Also, we posed the independence assumption among context types in most parts of this thesis except for Chapters 4 and 7, although this is clearly unrealistic in some cases as we showed in Chapter 4. Some feature selection methods incorporating mutual correlation among features [Ding and Peng, 2003] may be applicable for context selection.

There are still some issues to address concerning the supervised approaches we proposed in Chapter 7, because the current approach is only in its infancy. For example, although we used total correlation to build distributional features this time, the formalization requires further investigation. Even so, the proposed methods are definitely an important step toward even better synonym acquisition, because it enabled us to use various techniques proposed in the machine learning field. This will lead us to a new frontier of lexical knowledge acquisition, where the state-of-the-art techniques such as semi-supervised learning can be made the most of.

There are two major problems in distributional similarity field in general: *polysemy* and *relatedness*. A word can have multiple meanings, although the current similarity approaches are not capable of addressing this issue directly. Some recent work (e.g., [Snow et al., 2006]) have begun to deal with multiple senses of words. More and more polysemy-aware methods are to be expected in the future. As for the *relatedness* issue, distributional similarity is often critisized that it only extracts *related* words, not *synonyms*. This is a difficult problem to solve only by refining the similarity computation. Some researchers [Lin et al., 2003] resorted to the augmentation by syntactic patterns. Our proposed method in Chapter 7 also has the potential to deal with this problem only by replacing the training data.

Setting our future direction on the completely automatic construction of reliable lexical knowledge bases such as thesauri and ontologies, the approaches proposed here are to be applied to and integrated with various kinds of lexical knowledge acquisition methods in the future.

Appendix: List of the 100 Chosen LDV Words

action, advantage, anxiety, article, background, bank, blood, board, bunch, burial, business, call, care, case, cloud, cold, combination, comparison, competition, crowd, curse, curve, dance, delay, department, education, effect, faint, family, feeling, field, fold, footstep, green, ground, group, guide, guilt, habit, health, heaven, idea, imagination, instruction, interruption, job, joint, kiss, knowledge, laughter, lawyer, length, letter, manner, member, minister, mistake, model, moment, noise, nothing, official, opinion, party, payment, pet, piece, police, port, possession, pound, preparation, punishment, remark, reply, report, representative, result, rod, room, row, rule, shade, shore, side, sight, soul, spring, story, structure, sympathy, tail, tear, terror, thought, threat, victory, view, wing, worship

Acknowledgements

I would like to express my sincere gratitude to Associate Professor Katsuhiko Toyama, and Assistant Professor Yasuhiro Ogawa at Graduate School of Information Science, Nagoya University, who have been continuously assisting and guiding me with constant patience and eagerness. Also, I would like to thank Associate Professor Shigeki Matsubara, Associate Professor Yukiko Yamaguchi of Information Technology Center, Nagoya University, Dr. Kazue Sugino, Dr. Muhtar Mahsut, Professor Dr. Masatoshi Yoshikawa at Kyoto University for their kind support in constructive discussions and everyday life.

I would also like to thank Prof. Toshiki Sakabe and Prof. Kiyoshi Agusa for their useful comments and feedback on this thesis.

Outside and inside the laboratory, I was also given kind encouragement and support from my colleagues and lab-mates. They are too many to name, but I must mention Tomohiro Ohno and Koichiro Ryu, who have been kind and supportive in both research and daily life ever since I joined Toyama Group.

My internship experiences should definitely be remembered, though not directly related to the research work described in this thesis. My mentors Dr. Dekang Lin and Dr. Jun Wu at Google in Mountain View, CA, a colleague Masanori Harada, as well as other Japanese interns — Mizuki Oka, Tomoyuki Nanno, who now works at Google, and Daisuke Shimamoto are truly smart people and they were tremendously inspiring and supportive not only during the internship period but also the life after. Also, the internship experience at Microsoft Research in Redmond, WA, was greatly fruitful and it affected my research beyond my imagination. I cannot definitely thank my mentor Dr. Hisami Suzuki enough, who has been kindly guiding me even after the internship. The encouragement and support I received from other researchers in MSR NLP groups, namely, Takako Aikawa, Dmitriy Belenko, Chris Brockett, Jianfeng Gao, Christian König, and Chris Quirk, and other interns, especially Aria Haghighi and Hoifung Poon, was unmeasurable, too.

The work described in Chapter 7 would not have been possible without the help of Project Assistant Professor Nobuyuki Shimizu and Professor Hiroshi Nakagawa at Information Technology Center, University of Tokyo. I am especially grateful for the interesting discussion we had and support from Dr. Shimizu, whose knowledge and experience especially in machine learning field was indispensable for the work.

The MITO Exploratory Software Project, *"Serendi: A Location-Aware Social Networking Platform"* was also a very unique experience to me, and I would like to thank precious co-developers with whom I had very useful discussion, namely, Udana Ban-

dara at University of Tokyo, Yuto Imazeki at Keio University, and Junya Nakamura and Fumihiko Koyama at GOGA Inc.

Even outside the university, a number of researchers have given my work kind feedback and support. To name a few, I would like to thank Professor Kyoji Umemura at Toyohashi University of Technology, Mr. Mamoru Komachi at Nara Institute of Science and Technology, Dr. Manabu Sassano at Yahoo Japan Corporation, and Lecturer Eiji Aramaki at University of Tokyo. I especially appreciate Mr. Komachi's support and advice for virtually everything related to my research career, without whom I would not have had my great experience at Microsoft Research.

Lastly but definitely not least importantly, I would like to express my sincere appreciation to all my friends and families, who have been supporting and encouraging me not only during the difficult time of research, but also throughout my entire life.

Publications

Peer-reviewed Journal Papers

Related Chapters

Masato Hagiwara, Yasuhiro Ogawa, Katsuhiko Toyama. Supervised Synonym Acquisition Using Distributional Features and Syntactic Patterns. Journal of Natural Language Processing, (to appear). — Chapter 7

Masato Hagiwara, Yasuhiro Ogawa, Katsuhiko Toyama. A Comparative Study on Effective Context Selection for Distributional Similarity. Journal of Natural Language Processing, Volume 15, Number 5, pp. 119-150, 2008. — Chapter 6

Masato Hagiwara, Yasuhiro Ogawa, Katsuhiko Toyama. Effective Use of Indirect Dependency for Distributional Similarity. Journal of Natural Language Processing, Volume 15, Number 4, pp. 19-42, 2008. — Chapter 5

Peer-reviewed International Conference Papers

Nobuyuki Shimizu, Masato Hagiwara, Yasuhiro Ogawa, Katsuhiko Toyama and Hiroshi Nakagawa. Metric learning for synonym acquisition. In Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008), pp. 793-800, 2008. — Chapter 7

Masato Hagiwara. A Supervised Learning Approach to Automatic Synonym Identification based on Distributional Features. In Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL 2008), Student Research Workshop, pp. 1-6, 2008. — Chapter 7

Masato Hagiwara, Yasuhiro Ogawa, Katsuhiko Toyama. Context Feature Selection for Distributional Similarity. In Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP 2008), pp. 553-560, 2008. — Chapter 6

Masato Hagiwara, Yasuhiro Ogawa, Katsuhiko Toyama. Effective Proximity Distance for Word-Based Context. In Proceedings of the Seventh International Symposium on Natural Language Processing (SNLP2007), pp. 105-110, 2007. — Chapter 6

Masato Hagiwara, Yasuhiro Ogawa, Katsuhiko Toyama. Effectiveness of Indirect Dependency for Automatic Synonym Acquisition. In Proceedings of Contextual Information in Semantic Space Models: Beyond Words and Documents (CoSMo 2007), pp. 1 - 8, 2007. — Chapter 5

Masato Hagiwara, Yasuhiro Ogawa, Katsuhiko Toyama. Selection of Effective Contextual Information for Automatic Synonym Acquisition. In Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (COLING/ACL 2006), pp. 353 - 360, 2006. — Chapter 6

Masato Hagiwara, Yasuhiro Ogawa, Katsuhiko Toyama. PLSI Utilization for Automatic Thesaurus Construction. In Proceedings of the Second International Joint Conference on Natural Language Processing (IJCNLP-05), pp. 334 - 345, 2005. — Chapters 2, 3, 4

Peer-reviewed Domestic Conference Paper

萩原正人，小川泰弘，外山勝彦．類義語自動獲得における間接依存関係の有効性．言語処理学会第 13 回年次大会ワークショップ「言語的オントロジーの構築・連携・利用」論文集 pp. 43-46, 2007.
(Masato Hagiwara, Yasuhiro Ogawa, Katsuhiko Toyama. Effectiveness of Indirect Dependency for Automatic Synonym Acquisition. In Proceedings of the Workshop on Linguistic Ontologies in the 13th Annual Meeting of the Association for Natural Language Proceeding, pp. 43-46, 2008) — Chapter 5

Bibliography

[Bilenko and Mooney, 2003] Bilenko, M. and Mooney, R. J. (2003). Adaptive duplicate detection using learnable string similarity measures. In *Proc. of ACM SIGKDD 03*, pages 39–48.

[Bingham and Mannila, 2001] Bingham, E. and Mannila, H. (2001). Random projection in dimensionality reduction: applications to image and text data. In *Proc. of ACM SIGKDD 01*, pages 245–250.

[Blei et al., 2003] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

[Briscoe et al., 2002] Briscoe, T., Carroll, J., Graham, J., and Copestake, A. (2002). Relational evaluation schemes. In *Proc. of the Beyond PARSEVAL Workshop at the Third International Conference on Language Resources and Evaluation*, pages 4–8.

[Briscoe et al., 2006] Briscoe, T., Carroll, J., and Watson, R. (2006). The second release of the rasp system. In *Proc. of the COLING/ACL 06 Interactive Presentation Sessions*, pages 77–80.

[Budanitsky and Hirst, 2006] Budanitsky, A. and Hirst, G. (2006). Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32:13–47.

[Caruana et al., 2004] Caruana, R., Joachims, T., and Backstrom, L. (2004). Kdd-cup 2004: results and analysis. *ACM SIGKDD Explorations Newsletter*, 6.

[Chung and Lee, 2001] Chung, Y. M. and Lee, J. Y. (2001). A corpus-based approach to comparative evaluation of statistical term association measures. *Journal of the American Society for Information Science and Technology*, 52(4):283–296.

[Collins, 2002] Collins (2002). *Collins Cobuild Major New Edition CD-ROM*. HarperCollins Publishers.

[Connor and Roth, 2007] Connor, M. and Roth, D. (2007). Context sensitive paraphrasing with a single unsupervised classifier. In *Proc. of the European Conference on Machine Learning (ECML 07)*.

[Crouch and Yang, 1992] Crouch, C. J. and Yang, B. (1992). Experiments in automatic statistical thesaurus construction. In *Proc. of the 15th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR 92)*, pages 77–88.

[Curran and Moens, 2002a] Curran, J. R. and Moens, M. (2002a). Improvements in automatic thesaurus extraction. In *Proc. of ACL SIGLEX*, pages 231–238. In Workshop on Unsupervised Lexical Acquisition.

[Curran and Moens, 2002b] Curran, J. R. and Moens, M. (2002b). Scaling context space. In *Proc. of ACL 02*, pages 231–238.

[Davis and Dhillon, 2006] Davis, J. V. and Dhillon, I. S. (2006). Differential entropic clustering of multivariate gaussians. In *Neural Information Processing Systems (NIPS) 19*, pages 337–344.

[Davis et al., 2007] Davis, J. V., Kulis, B., Jain, P., Sra, S., and Dhillon, I. S. (2007). Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning (ICML 07)*, pages 209–216.

[Deerwester et al., 1990] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

[Ding and Peng, 2003] Ding, C. and Peng, H. (2003). Minimum redundancy feature selection from microarray gene expression data. In *Proc. of the IEEE Computer Society Conference on Bioinformatics*, pages 523–528.

[Fellbaum, 1998] Fellbaum, C. (1998). *WordNet: an electronic lexical database*. MIT Press.

[Gamallo et al., 2001] Gamallo, P., Gasperin, C., Agustini, A., and Lopes, G. P. (2001). Syntactic-based methods for measuring word similarity. *Text, Speech, and Discourse (TSD 01), Lecture Notes in Computer Science*, 2166:116–125.

[Geffet and Dagan, 2004] Geffet, M. and Dagan, I. (2004). Feature vector quality and distributional similarity. In *Proc. of COLING 04*, pages 247–253.

[Globerson and Roweis, 2006] Globerson, A. and Roweis, S. (2006). Metric learning by collapsing classes. In *Advances in Neural Information Processing Systems (NIPS) 18*, pages 451–458.

[Goldberger et al., 2004] Goldberger, J., Roweis, S., Hinton, G., and Salakhutdinov, R. (2004). Neighborhood component analysis. In *Advances in Neural Information Processing Systems (NIPS) 17*, pages 512–520.

[Grefenstette, 1994] Grefenstette, G. (1994). *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publisher.

[Hagiwara et al., 2006] Hagiwara, M., Ogawa, Y., and Toyama, K. (2006). Selection of effective contextual information for automatic synonym acquisition. In *Proc. of COLING/ACL 06*, pages 353–360.

[Harris and (ed.), 1985] Harris, Z. and (ed.), J. J. K. (1985). *The Philosophy of Linguistics*, pages 26–47. Oxford University Press.

[Hastie and Tibshirani, 1996] Hastie, T. and Tibshirani, R. (1996). Discriminant adaptive nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(6):607–616.

[Hearst, 1993] Hearst, M. A. (1993). Automatic acquisition of hyponyms from large text corpora. In *Proc. of COLING 92*, pages 539–545.

[Hindle, 1990] Hindle, D. (1990). Noun classification from predicate-argument structures. In *Proc. of ACL 90*, pages 268–275.

[Hirst and St-Onge, 1998] Hirst, G. and St-Onge, D. (1998). Lexical chains as representations of context for the detection and correction of malapropisms. In *Christiane Fellbaum ed. WordNet: An electronic lexical database*, pages 305–332. MIT Press.

[Hofmann, 1999] Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proc. of the 22nd International Conference on Research and Development in Information Retrieval (SIGIR 99)*, pages 50–57.

[Hofmann, 2001] Hofmann, T. (2001). Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42:177–196.

[Hofmann et al., 1999] Hofmann, T., Puzicha, J., and Jordan, M. I. (1999). Learning from dyadic data. In *Proceedings of Advances in Neural Information Processing Systems (NIPS) 11*, pages 466–472.

[Jiang and Conrath, 1997] Jiang, J. J. and Conrath, D. W. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. In *Proc. of International Conference on Research in Computational Linguistics*, pages 19–33.

[Jing and Croft, 1994] Jing, Y. and Croft, B. (1994). An association thesaurus for information retrieval. In *Proc. of RIAO(Recherche d'Informations Assistée par Ordinateur) 1994*, pages 146–160.

[Joachims, 1999] Joachims, T. (1999). *Making Large-Scale SVM Learning Practical*. MIT-Press.

[Kandola et al., 2002] Kandola, J., Shawe-Taylor, J., and Cristianini, N. (2002). Learning semantic similarity. In *Neural Information Processing Systems (NIPS) 15*, pages 657–664.

[Kita and Tsujii, 1999] Kita, K. and Tsujii, J. (1999). *Computation and Language Volume 4: Probabilistic Language Model*. University of Tokyo Press.

[Kojima and Ito, 1995] Kojima, H. and Ito, A. (1995). Adaptive scaling of a semantic space. In *IPSJ SIGNotes Natural Language, NL108-13*, pages 81–88.

[Lee, 1999] Lee, L. (1999). Measures of distributional similarity. In *Proc. of ACL 99*, pages 25–32.

[Lee, 2001] Lee, L. (2001). On the effectiveness of the skew divergence for statistical language analysis. In *Artificial Intelligence and Statistics 2001*, pages 65–72.

[Lenat, 1995] Lenat, D. (1995). Cyc: A large-scale investment in knowledge infrastructure. *Cmmunications of the ACM*, 38(11).

[Lin, 1998a] Lin, D. (1998a). Automatic retrieval and clustering of similar words. In *Proc. of COLING/ACL 98*, pages 786–774.

[Lin, 1998b] Lin, D. (1998b). Information-theoretic definition of similarity. In *Proc. of ICML 98*, pages 294–304.

[Lin and Pantel, 2001] Lin, D. and Pantel, P. (2001). Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):343–360.

[Lin et al., 2003] Lin, D., Zhao, S., Qin, L., and Zhou, M. (2003). Identifying synonyms among distributionally similar words. In *Proc. of IJCAI 03*, pages 1492–1493.

[Lin, 1991] Lin, J. (1991). Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):140–151.

[Lowe and McDonald, 2000] Lowe, W. and McDonald, S. (2000). The direct route: Mediated priming in semantic space. In *Proc. of the 22nd Annual Conference of the Cognitive Science Society*, pages 675–680.

[Mirkin et al., 2006] Mirkin, S., Dagan, I., and Geffet, M. (2006). Integrating pattern-based and distributional similarity methods for lexical entailment acquisition. In *Proc. of COLING/ACL 06*, pages 579–586.

[Mochihashi and Matsumoto, 2002] Mochihashi, D. and Matsumoto, Y. (2002). Probabilistic representation of meanings. *IPSJ SIGNotes Natural Language, 2002-NL-147*, pages 77–84.

[Ng and Lee, 1996] Ng, H. T. and Lee, H. B. (1996). Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. In *Proc. of ACL 96*, pages 40–47.

[Kokken, 2004] The National Institute of Japanese Language (2004). *Bunruigoihyo*. Dainippontosho.

[Editors of the American Heritage Dictionary, 1995] Editors of the American Heritage Dictionary (1995). *Roget's II: The New Thesaurus, 3rd ed.* Houghton Mifflin.

[Oyama and Manning, 2004] Oyama, S. and Manning, C. D. (2004). Using feature conjunctions across examples for learning pairwise classifiers. In *Proc. of ECML 04*, pages 322–333.

[Pado and Lapata, 2007] Pado, S. and Lapata, M. (2007). Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

[Pantel and Pennacchiotti, 2006] Pantel, P. and Pennacchiotti, M. (2006). Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proc. of COLING/ACL 06*, pages 113–120.

[Pedersen et al., 2004] Pedersen, T., Patwardhan, S., and Michelizzi, J. (2004). Wordnet::similarity - measuring the relatedness of concepts. In *Proc. of NAACL 04*, pages 38–41.

[Pereira et al., 1993] Pereira, F., Tishby, N., and Lee, L. (1993). Distributional clustering of english word. In *Proc. of ACL 93*, pages 183–190.

[Picard, 1999] Picard, J. (1999). Finding content-bearing terms using term similarities. In *Proc. of EACL 99*, pages 241–244.

[Ravichandran et al., 2005] Ravichandran, D., Pantel, P., and Hovy, E. (2005). Randomized algorithms and nlp: using locality sensitive hash function for high speed noun clustering. In *Proc. of the 43rd Annual Meeting on Association for Computational Linguistics (ACL 05)*, pages 622–629.

[Riloff and Shepherd, 1997] Riloff, E. and Shepherd, J. (1997). A corpus-based approach for building semantic lexicons. In *Proc. of EMNLP 97*, pages 117–124.

[Roark and Charniak, 1998] Roark, B. and Charniak, E. (1998). Noun phrase cooccurrence statistics for semi-automatic lexicon construction. In *Proc. of COLING/ACL 98*, pages 1110–1116.

[Ruge, 1997] Ruge, G. (1997). Automatic detection of thesaurus relations for information retrieval applications. *Foundations of Computer Science: Potential - Theory - Cognition, Lecture Notes in Computer Science*, 1337:499–506.

[Sakaki et al., 2007] Sakaki, T., Matsuo, Y., Uchiyama, K., and Ishizuka, M. (2007). Construction of related terms thesauri from the web. *Journal of Natural Language Procesing*, 14(2):3–31.

[Schultz and Joachims, 2003] Schultz, M. and Joachims, T. (2003). Learning a distance metric from relative comparisons. In *Advances in Neural Information Processing Systems (NIPS) 16*.

[Shalev-shwartz et al., 2004] Shalev-shwartz, S., Singer, Y., and Ng, A. Y. (2004). Online and batch learning of pseudo-metrics. In *Proceedings of the International Conference on Machine Learning (ICML 04)*, pages 94–101.

[Snow et al., 2006] Snow, R., Jurafsky, D., and Ng, A. Y. (2006). Semantic taxonomy induction from heterogenous evidence. In *Proc. of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL (COLING/ACL 06)*, pages 801–808.

[Snow et al., 2004] Snow, R., Jurafsly, D., and Ng, A. Y. (2004). Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems (NIPS) 17*, pages 1297–1304.

[Tanimoto, 1957] Tanimoto, T. T. (1957). In *IBM Internal Report 17th Nov. 1957*.

[Thelen and Riloff, 2002] Thelen, M. and Riloff, E. (2002). A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proc. of EMNLP 02*, pages 214–221.

[Ueda and Nakano, 2002] Ueda, N. and Nakano, R. (2002). Deterministic annealing em algorithm. *Neural Networks*, 11:271–282.

[Weeds and Weir, 2003] Weeds, J. and Weir, D. (2003). A general framework for distributional similarity. In *Proc. of EMNLP 03*, pages 81–88.

[Weeds et al., 2004] Weeds, J., Weir, D., and McCarthy, D. (2004). Characterising measures of lexical distributional similarity. In *Proc. of COLING 04*, pages 1015–1021.

[Weinberger et al., 2006] Weinberger, K. Q., Blitzer, J., and Saul, L. K. (2006). Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems (NIPS) 18*, pages 1473–1480.

[Wilbur and Sirotkin, 1992] Wilbur, J. and Sirotkin, K. (1992). The automatic identification of stop words. *Journal of Information Science*, 18(1):45–55.

[Xing et al., 2003] Xing, E. P., Ng, A. Y., Jordan, M. I., and Russell, S. (2003). Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems (NIPS) 15*, pages 505–512.

[Yang and Pedersen, 1997] Yang, Y. and Pedersen, J. O. (1997). A comperative study on feature selection in text categorization. In *Proc. of ICML 97*, pages 412–420.

[Yang and Wilbur, 1996] Yang, Y. and Wilbur, J. (1996). Using corpus statistics to remove redundant words in text categorization. *Journal of the American Society for Information Science*, 47(5):357–369.

[Yu and Agichtein, 2003] Yu, H. and Agichtein, E. (2003). Extracting synonymous gene and protein terms from biological literature. *Bioinfomatics*, 19(1):i340–i349.