Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

Supervised Synonym Acquisition Using Distributional Features and Syntactic Patterns

Masato Hagiwara†, Yasuhiro Ogawa† and Katsuhiko Toyama†

Distributional similarity has been widely used to capture the semantic relatedness of words in many NLP tasks. However, parameters such as similarity measures must be manually tuned to make distributional similarity work effectively. To address this problem, we propose a novel approach to synonym identification based on supervised learning and *distributional features*, which correspond to the commonality of individual context types shared by word pairs. This approach also enables the integration with *pattern-based features*. In our experiment, we have built and compared eight synonym classifiers, and showed a drastic performance increase of over 60% on F-1 measure, compared to the conventional similarity-based classification. Distributional features that we have proposed are better in classifying synonyms than the conventional *common features*, while the pattern-based features have appeared almost redundant.

Key Words: *Distributional Similarity, Synonym Acquisition, Distributional Features, Syntactic Patterns, Pairwise Classification*

1 Introduction

Semantic similarity of words is one of the most important lexical knowledge for natural language processing, having a broad range of applications such as query expansion for information retrieval (Crouch and Yang 1992; Jing and Croft 1994), word sense disambiguation (Ng and Lee 1996), and thesaurus construction (Kojima and Ito 1995; Grefenstette 1994). To measure semantic relatedness of words, a concept called *distributional similarity* has been widely used. It measures the similarity of two words by the commonality of contexts the words share, based on the distributional hypothesis (Harris and Katz 1985), which states that semantically similar words tend to share similar contexts. Conversely, two words that share a certain amount of similar contexts are likely to be semantically similar.

A number of researches which utilized distributional similarity have been conducted, including (Hindle 1990; Lin 1998; Geffet and Dagan 2004) and many others. Although they have been successful in acquiring related words, this involves various parameters such as similarity measures and weight functions. As Weeds et al. (Weeds, Weir, and McCarthy 2004) pointed out, “it is not

† Graduate School of Information Science, Nagoya University.

Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

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, but 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 be easily converted to features, and machine learning approach can be used to discriminate the relations that the words have. Snow et al. (Snow, Jurafsly, and Ng 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, Zhao, Qin, and Zhou 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, Dagan, and Geffet 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 will be essential, as pattern-based lexical knowledge acquisition is enhanced with the supervised learning.

In this paper, 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 paper is three-fold. Firstly, we re-formalize synonym acquisition as a classification problem: one which classifies *word pairs* into synonym/non-synonym classes, without

Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

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 finally 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.

The rest of this paper is organized as follows: after introducing related work in Section 2, we briefly review the conventional distributional similarity model in Section 3. In Section 4, 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 5. Finally we build eight types of synonym classifiers, and compare their performances in Section 6. Section 7 concludes this paper.

2 Related Work

2.1 Distributional Similarity

As we mentioned in Section 1, there are a number of researches regarding distributional similarity (Hindle 1990; Lin 1998; Geffet and Dagan 2004). However, these methods are all unsupervised approaches, thus appropriate weights and similarity measures have to be fixed beforehand, and they 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. 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 metaparameters were optimized on a development set, and 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 do in this study.

Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

2.2 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 filed, 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." 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, but it was designed for binary relation extraction, and it 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.

2.3 Classification Approach

Supervised classification 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, and 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 this paper.

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 a SVM classifier, using contextual patterns (i.e., connecting string, as done in *Espresso*).

Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

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, and it does not directly compete with our method—the feature construction introduced here can be used together with their framework.

3 Distributional Similarity

In this section, we briefly review the conventional approach to synonym acquisition, i.e., the distributional similarity model, where the similarity between two words is defined as the degree of commonality of the context that two words share. The computation of distributional similarity usually consists of three elements: (1) context extraction, (2) co-occurrence weighting, and (3) similarity calculation, each of which is detailed in the following.

3.1 Context Extraction

In the first step of the model, the contexts of words are extracted from corpora, based on the previously mentioned distributional hypothesis. 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 1998), and dependency path (Lin and Pantel 2001; Pado and Lapata 2007). Among these we adopt dependency structure as the context of words since it is the most widely used and it is shown to be well-performing contextual information in the past studies, although the approach or discussion described in this paper is not limited to specific contextual information of choice.

In this paper the sophisticated parser RASP Toolkit 2 (Briscoe, Carroll, and Watson 2006) is utilized to extract this kind of word relations. We use the following example for illustration purposes:

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

RASP outputs the extracted dependency structure as n-ary relations as follows, whose graphical representation is shown in Figure 1:

```
(ncsubj have library _)
(dobj have collection)
(det collection a)
(ncmod _ collection large)
```

Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

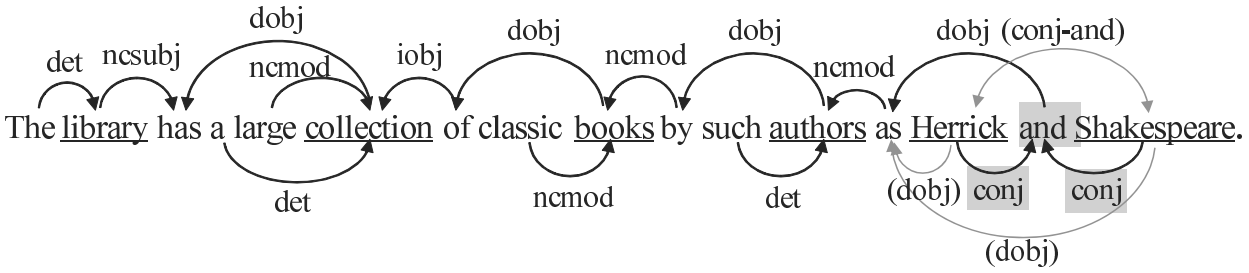

Fig. 1 Dependency structure of the example sentence, along with conjunction shortcuts (dotted lines).

```
(iobj collection of)
(dobj of book)
(ncmod _ book by)
(dobj by author)
(det author such)
(ncmod _ author as)
...
```

While the RASP outputs are n-ary relations in general, what we need here is co-occurrences of words and their contexts, so we extract the set of co-occurrences of stemmed words and 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)
...
```

Summing all these up produces the raw co-occurrence count $N(w, c)$ of word w and context c.

Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

3.2 Co-occurrence Weighting

As the second step, the context types extracted above can be weighted in various ways, including tf.idf and χ^2 statistics, and many others are proposed (Curran and Moens 2002a; Weeds et al. 2004), but in this paper we employ *pointwise mutual information* (PMI) because it performed the best among various weighting functions in a preliminary experiment. Using PMI, the weight of the co-occurrence (w_i, c_j) of a word w_i and a context type c_j is defined as:

$$\mathrm{wgt}(w_i, c_j) = \mathrm{PMI}(w_i, c_j) = \log \frac{P(w_i, c_j)}{P(w_i)P(c_j)}, \tag{1}$$

where $P(w_i, c_j)$ is the probability of the word w_i co-occurring with the context c_j. This probability is empirically calculated as $P(w_i, c_j) = N(w_i, c_j) / \sum_{w,c} N(w, c)$ using the co-occurrence frequency $N(w_i, c_j)$ obtained from corpora. After this weighting step, each word can be represented as a vector whose elements correspond to the weights:

$$\mathbf{w_i} = [\mathrm{wgt}(w_i, c_1)\ \mathrm{wgt}(w_i, c_2)\ \ldots\ \mathrm{wgt}(w_i, c_M)]^T, \tag{2}$$

where M is the number of context types.

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

3.3 Similarity Calculation

In the third step of this model, the similarity between words are calculated using some similarity/distance measures. There are a variety of similarity measures proposed, such as cosine similarity, Jaccard coefficient, and confusion probability (Curran and Moens 2002a; Weeds et al. 2004). We employed cosine similarity, again, considering the preliminary result. It defines the similarity of word w_1 and w_2 as:

$$f^S(w_1, w_2) = \frac{\mathbf{w}_1 \cdot \mathbf{w}_2}{||\mathbf{w}_1|| \cdot ||\mathbf{w}_2||} = \frac{\sum_j \mathrm{wgt}(w_1, c_j) \cdot \mathrm{wgt}(w_2, c_j)}{\sqrt{\sum_j \mathrm{wgt}(w_1, c_j)^2} \cdot \sqrt{\sum_j \mathrm{wgt}(w_2, c_j)^2}} \tag{3}$$

In the synonym detection task, a threshold can be set on the similarity, and the word pairs whose similarities are above this threshold are marked as synonym pairs. How to optimally set this threshold is described in Section 6.2.

Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

4 Pairwise Classification

This section describes a different approach to synonym acquisition, i.e., pairwise synonym classification. The problem formalization is as follows. Firstly, we deal with word pairs, instead of individual words, as the instances of classification. We then prepare features $f_1, ..., f_M$ constructed as described below. The target class y is defined such that $y = 1$ if the pair is true synonyms, i.e., similar words not including related words, hypernyms/hyponyms, antonyms, etc., and $y = 0$ if not. Pairs with $y = 1$ are called positive examples, while pairs with $y = 0$ are called negative examples in this paper. With these defined, the problem is formalized as pairwise binary classification which assigns the classes $y \in \{0, 1\}$ to the word pairs based on the features $f_1, ..., f_M$ of the pair.

We adopt two schemes to construct features in the classification task: *common features* and *distributional features*, and the detail is given below.

4.1 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:

$$\mathbf{x} = [x_1\ x_2\ \ldots\ x_M]^T, \qquad x_j = \mathrm{wgt}(x, c_j), \tag{4}$$

$$\mathbf{z} = [z_1\ z_2\ \ldots\ z_M]^T, \qquad z_j = \mathrm{wgt}(z, c_j). \tag{5}$$

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

Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

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 feature vector $\mathbf{p}_{\text{cfeat}}$ for the pair $p = (x, z)$ as:

$$f_j^C(x, z) = \text{wgt}(x, c_j) \cdot \text{wgt}(z, c_j) \tag{6}$$

$$\mathbf{p}_{\text{cfeat}} = [f_1^C(x, z)\ f_2^C(x, z)\ \ldots\ f_M^C(x, z)]^T. \tag{7}$$

This feature construction can be interpreted as follows—instead of using a single, integrated value of the summation in Equation (3), 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.

4.2 Distributional Features

The pair features do not necessarily 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:

$$\mathbf{p}_{\text{dfeat}} = [f_1^D(x, z)\ f_2^D(x, z)\ \ldots\ f_M^D(x, z)]^T \tag{8}$$

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, and we adopt this as the feature value:

$$f_j^D(x, z) = \log \frac{P(x, z, c_j)}{P(x)P(z)P(c_j)}. \tag{9}$$

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:

$$f_j^D(x, z) = \text{PMI}(x, c_j) + \text{PMI}(z, c_j), \tag{10}$$

where the value of $\text{PMI}(x, c_j)$, which is also the weights $\text{wgt}(x, c_j)$ assigned for distributional similarity, is calculated as shown in Equation (1). Note that the feature value $f_j^D(x, z)$ in Equation (9) is finite only when the context c_j co-occurs with both x and z. In practice, however, we used the definition in Equation (10) with the independence assumption, and the value of $f_j^D(x, z)$ is

Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

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 (6), while the latter uses the sum. Therefore, when the weights are PMI, Equation (8) can be rewritten as:

$$\mathbf{p}_{\mathrm{dfeat}} = [x_1 + z_1 \quad x_2 + z_2 \quad \ldots \quad x_M + z_M]^T. \tag{11}$$

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

5 Pattern-based Features

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

5.1 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; 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.

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 paper 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 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.

Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

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. 1 as dotted lines.

Snow et al. also proposed *satellite links*, i.e., single depending nodes attached at the end of paths, but 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 PoS tags of RASP2 outputs, and all the replacing operations we employed are listed in Table 1. For example, if we have `Herrick:dobj:as` as a pattern and the PoS tag of the word `Herrick` is `NP1`, the pattern is replaced with `NP:dobj:as`.

Table 1 Word generalization operations to syntactic patterns

CLAWS7 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.

Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

5.2 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:

$$f_k^P(w_1, w_2) = P(p_k|w_1, w_2) = \frac{n'(w_1, w_2, p_k)}{n'(w_1, w_2)}. \tag{12}$$

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 paper, 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.

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.

6 Experiments

6.1 Synonym Classifiers

Now that we have all features available, we build eight synonym classifiers and compare their performances. All the classifiers built are listed in Table 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

Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

Table 2 Synonym classifiers to compare

Name	Description	Features Used
DSIM	Distributional Similarity (unsupervised)	f^S
PAT	Pattern-based Features	$f^P_1, ..., f^P_K$
DSIM-PAT	DSIM and Pattern-based Features	$f^S, f^P_1, ..., f^P_K$
CFEAT	Common Features	$f^C_1, ..., f^C_M$
CFEAT-PAT	Common and Pattern-based Features	$f^C_1, ..., f^C_M, f^P_1, ..., f^P_K$
DFEAT	Distributional Features	$f^D_1, ..., f^D_M$
DFEAT-PAT	Distributional and Pattern-based Features	$f^D_1, ..., f^D_M, f^P_1, ..., f^P_K$
ALL	All Features	$f^S, f^P_1, ..., f^P_K, f^C_1, ..., f^C_M, f^D_1, ..., f^D_M$

f^S: distributional similarity, f^C_j: common features, f^D_j: distributional features, f^P_k: pattern-based features. M and K denote the numbers of context types and patterns, respectively.

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 6.2.

The remaining seven classifiers are all SVM-trained, supervised methods. PAT uses only pattern-based features, and it is essentially the same as Snow et al.'s method (Snow et al. 2004). DSIM-PAT is a semi-integrated classifier, and it 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 paper.

6.2 Experimental Settings

6.2.1 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. More specifically, any words w such that $\sum_c N(w,c) < \theta_w$ and any context types c such that $\sum_w N(w,c) < \theta_c$ were removed. The first threshold, the word cut-off frequency θ_w, is relatively easy to fix, because it does not directly affect the classification. The second threshold, the context cut-off frequency, is less easy to handle, because it directly affects the classification performance. We empirically investigated the performance change, and a preliminary experiment indicated that the performance actually *increased* almost linearly as

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

Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

we increased θ_c until $\theta_c = 5$, but after this point on, it began to decrease. Considering this result, we set $\theta_w = 5$ and $\theta_c = 5$, where the performance change was kept within 2% range while the context types are cut-down by more than 70% (from 145,358 to 39,750 in number).

We then applied DF (document frequency) thresholding to further reduce the dimensionality of vectors. The context types with the lowest values of DF were removed until 10% of the original contexts remained. This feature selection is shown to keep the performance loss at minimum (Hagiwara, Ogawa, and Toyama 2008), and it reduced the performance only by 8% in our case. As a result, this process left a total of 3,975 context types, i.e., feature dimensionality.

We also applied the feature selection 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.

6.2.2 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, three existing thesauri were 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 the head word is looked up as a noun is used as the answer set, except for words marked as "idiom," "informal," "slang" and phrases comprised of two or more words. 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 one for testing. SVM^{light} (Joachims 1999) was adopted for machine learning, and 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

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

Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

kernel, and the cost-factor j of 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.

Here the example set construction stated above seems greatly dependent on the LDV. Ideally, the example set should be randomly created out of a unlimited set of words. However, the open test described in Section 6.4 shows that the supervised classifiers perform well even for non-LDV words, suggesting that LDV-trained models are not dependent on LDV but general enough to be applied to out-of-LDV synonym classification.

6.3 Performance Comparison

The performances, i.e., precision, recall, and F-1 measure, of the seven classifiers were evaluated and shown in Table 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, and 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

Table 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%**

Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

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, and 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, it did not achieve as high recall as CFEAT/DFEAT did, but 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, and it is computationally extensive. Therefore, the trade-off between computational cost and performance gain should be taken into consideration in practice.

[3] We also tried the CFEAT-DFEAT classifier, i.e., ALL without PAT and DSIM, but 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.

Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

6.4 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. The task here is actually to *rank* synonyms, not to classify them, and 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 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, and this 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 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

Table 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 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

Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

Table 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

out that this was the most common case. Here even DSIM performs quite well, and so do the others. The result of CFEAT seems to include the fewer number of relevant words, but 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 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 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, and 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.

6.5 Error Anaysis

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.

Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

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*), and 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 wasn't 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, but 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.

Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

6.6 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. There can be a number of methods to rank or select features for machine learning, but 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 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

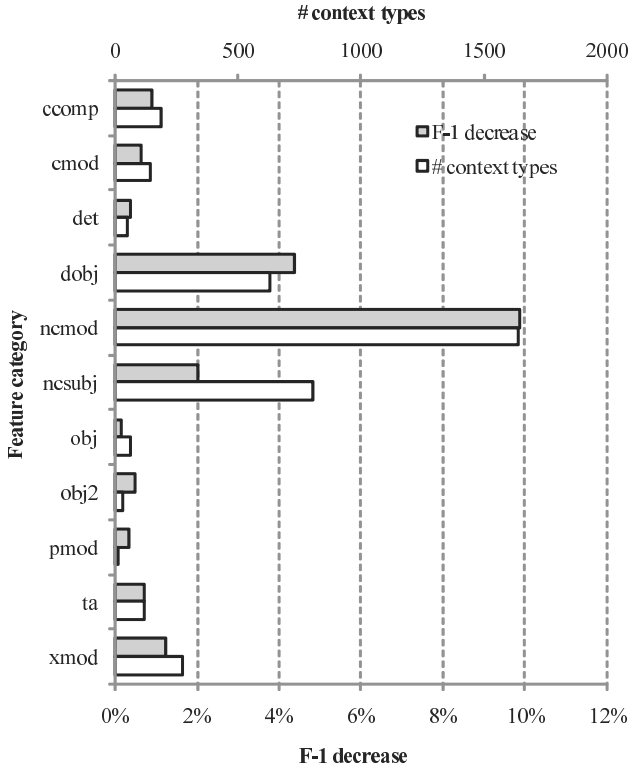

Fig. 2 Feature categories and their contributions

Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

Table 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

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, and 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, and 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, 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 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.

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

Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

7 Conclusion

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

The impact of this study is that it makes unnecessary to carefully choose similarity measures such as cosine and Jaccard's. Instead, features 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, and how they affect the synonym acquisition performance should be investigated in the future.

Reference

Bilenko, M. and Mooney, R. J. (2003). "Adaptive Duplicate Detection Using Learnable String Similarity Measures." In *Proc. of ACM SIGKDD*, pp. 39–48.

Briscoe, T., Carroll, J., and Watson, R. (2006). "The Second Release of the RASP System." In *Proc. of the COLING/ACL 2006 Interactive Presentation Sessions*, pp. 77–80.

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

Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

Connor, M. and Roth, D. (2007). "Context Sensitive Paraphrasing with a Single Unsupervised Classifier." In *Proc. of the European Conference on Machine Learning (ECML)*.

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*, pp. 77–88.

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

Curran, J. R. and Moens, M. (2002b). "Scaling Context Space." In *Proc. of ACL 2002*, pp. 231–238.

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

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

Geffet, M. and Dagan, I. (2004). "Feature Vector Quality and Distributional Similarity." In *Proc. of COLING 2004*, pp. 247–253.

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

Hagiwara, M., Ogawa, Y., and Toyama, K. (2008). "Context Feature Selection for Distributional Similarity." In *Proc. of IJCNLP 2008*, pp. 553–560.

Harris, Z. and Katz, J. J. (1985). *The Philosophy of Linguistics*, pp. 26–47. Oxford University Press.

Hearst, M. A. (1993). "Automatic Acquisition of Hyponyms from Large Text Corpora." In *Proc. of COLING 1992*, pp. 539–545.

Hindle, D. (1990). "Noun classification from predicate-argument structures." In *Proc. of ACL 1990*, pp. 268–275.

Jing, Y. and Croft, B. (1994). "An Association Thesaurus for Information Retrieval." In *Proc. of RIAO(Recherche d'Informations Assistée par Ordinateur) 1994*, pp. 146–160.

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

Kojima, H. and Ito, A. (1995). "Adaptive Scaling of a Semantic Space." In *IPSJ SIGNotes Natural Language, NL108-13*, pp. 81–88.

Lee, L. (1999). "Measures of Distributional Similarity." In *Proc. of ACL 1999*, pp. 25–32.

Lin, D. (1998). "Automatic retrieval and clustering of similar words." In *Proc. of COLING/ACL 1998*, pp. 786–774.

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

Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

Lin, D., Zhao, S., Qin, L., and Zhou, M. (2003). "Identifying Synonyms among Distributionally Similar Words." In *Proc. of IJCAL 2003*, pp. 1492–1493.

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*, pp. 675–680.

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

Ng, H. T. and Lee, H. B. (1996). "Integrating Multiple Knowledge Sources to Disambiguate Word Sense: An Exemplar-Based Approach." In *Proc. of ACL 1996*, pp. 40–47.

Oyama, S. and Manning, C. D. (2004). "Using Feature Conjunctions across Examples for Learning Pairwise Classifiers." In *Proc. of ECML 2004*, pp. 322–333.

Pado, S. and Lapata, M. (2007). "Dependency-Based Construction of Semantic Space Models." *Computational Linguistics*, **33** (2), pp. 161–199.

Pantel, P. and Pennacchiotti, M. (2006). "Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations." In *Proc. of COLING/ACL 2006*, pp. 113–120.

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

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

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**, pp. 499–506.

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*, pp. 1297–1304.

Thelen, M. and Riloff, E. (2002). "A Bootstrapping Method for Learning Semantic Lexicons using Extraction Pattern Contexts." In *Proc. of EMNLP 2002*, pp. 214–221.

Weeds, J. and Weir, D. (2003). "A General Framework for Distributional Similarity." In *Proc. of EMNLP 2003*, pp. 81–88.

Weeds, J., Weir, D., and McCarthy, D. (2004). "Characterising Measures of Lexical Distributional Similarity." In *Proc. of COLING 2004*, pp. 1015–1021.

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

Information and Media Technologies 4(2) : 558–582 (2009)
reprinted from: Journal of Natural Language Processing 16(2) : 59–83 (2009)

Masato Hagiwara: Masto Hagiwara received his Master's Degree of Information Science from Nagoya University in 2006, after skipping his fourth year of undergraduate. He is currently a Ph.D candidate, and his research interests include statistical natural language processing, especially lexical knowledge acquisition and automatic thesaurus construction.

Yasuhiro Ogawa: Yasuhiro Ogawa received his Bachelor's Degree, Master's Degree, and Ph. D. Degree of Engineering in 1995, 1997, and 2000, respectively. He is currently an assistant professor at the Graduate School of Information Science at Nagoya University. His research interests include Japanese morphological analysis, Japanese-Uighur machine translation, legal informatics, and e-legislation.

Katsuhiko Toyama: Katsuhiko Toyama received his Bachelor's Degree, Master's Degree, and Ph. D. Degree of Engineering in 1984, 1986, and 1989, respectively. He is currently an associate professor at the Graduate School of Information Science at Nagoya University. His research interests include natural language processing, especially legal informatics and e-legislation, as well as logical knowledge representation.

(Received September 17, 2008)
(Revised November 19, 2008)
(Accepted November 25, 2008)