Machine Translation **16:** 89–108, 2001.
© 2002 *Kluwer Academic Publishers. Printed in the Netherlands.*

Collocation Dictionary Optimization Using WordNet and *k*-Nearest Neighbor Learning ⋆

YUSEOP KIM, BYOUNG-TAK ZHANG and YUNG TAEK KIM
Department of Computer Engineering, Seoul National University, Seoul 151-742, Korea
E-mail: {yskim@nova, btzhang@comp, ytkim@comp}.snu.ac.kr

Abstract. In machine translation, collocation dictionaries are important for selecting accurate target words. However, if the dictionary size is too large it can decrease the efficiency of translation. This paper presents a method to develop a compact collocation dictionary for transitive verb–object pairs in English–Korean machine translation without losing translation accuracy. We use WordNet to calculate the semantic distance between words, and *k*-nearest neighbor learning to select the translations. The entries in the dictionary are minimized to balance the trade-off between translation accuracy and time. We have performed several experiments on a selected set of verbs extracted from a raw corpus of over 3 million words. The results show that in real-time translation environments the size of a collocation dictionary can be reduced up to 40% of its original size without significant decrease in its accuracy.

Key words: collocation, dictionary optimization, *k*-nearest neighbor learning, semantic distance, WordNet.

1. Introduction

Collocation information is important for improving accuracy in Machine Translation (MT) (Dagan and Itai, 1994). This information is usually stored in the form of a dictionary. The size of the collocation dictionary affects accuracy and time taken for translation. As the size of a collocation dictionary grows, translations tend to be more accurate, but the time required for translation tends to increase. Hence, the dictionary size should be tuned to balance the trade-off between translation accuracy and the time taken.

The optimization of dictionary size is generally linked with the problem of data sparseness. As the number of entries in the dictionary decreases, the frequency of direct matching to the dictionary also decreases resulting in low statistical values of collocation information. Several authors have proposed methods for resolving the problem of data sparseness or unregistered words. Most of these methods provide a mechanism for finding alternative words for a specific word when it does not appear in the corpora. Existing methods can be broadly divided into two categories: similarity-based and thesaurus-based approaches.

⋆ All trademarks are hereby acknowledged.

The similarity-based method has been proposed for disambiguating words by Karov and Edelman (1998) and Dagan et al. (1997, 1999). In this approach each word has a list of similar words and similarity in their values. The most similar word to the given word is chosen as the target word. The method has the advantage that it builds domain-dependent collocation information easily. However, it requires huge computations during the similarity learning phase and some additional certification process is required to handle multiple senses for an unseen word. This helps in resolving the problem only when the words that have information value are present in the training data.

Yarowsky (1992) presents a thesaurus-based method for word-sense disambiguation. He used *Roget's Thesaurus* to classify words into several categories. The alternative words are then chosen from the words in the category of a given word. Resnik (1995) describes a similar method in which WordNet (Miller et al., 1990) is used as a thesaurus for generating candidate words. Thesaurus-based methods have the advantage of determining the meaning of unseen words, too. However, word categorization can cause some problems since the features of each member in a category are not identical. For example, if words *green* and *red* are included in a category 'color', problems occur with words such as *green apple* and *red apple*, which cannot be treated as the same.

In this paper, we present a method for target word selection that combines the advantages of both the similarity-based and thesaurus-based methods. We use WordNet to resolve the data sparseness problem and/or the problem of unseen words in MT to translate transitive verbs. By using WordNet, we can select an alternative word unseen in the training data. In contrast to Resnik's approach where all the words under a synset (or a node in WordNet) are considered as having similar value, we calculate the semantic distance of every pair of the words in WordNet. Using WordNet and a semantic distance calculation, we can reduce the collocation dictionary size. We introduce a dictionary optimization method that is based on k-nearest neighbor (k-NN) learning (Aha et al., 1991; Mitchell, 1997). Dictionary optimization involves supervised learning. The training data consists of verb–object pairs extracted from a corpus of the source language for translation.

The effectiveness of the above method was tested on the English–Korean MT system E-Tran2001 (SNU, 2001). Experiments have been performed on the *Wall Street Journal* raw corpus for about 3 million words. We obtained a translation accuracy of 88% when applied to the verb *build*, even when the collocation dictionary size was reduced up to 40% from the original. We also attained a translation accuracy of 91% on the core translation of unregistered object nouns as default meaning.

The paper is organized as follows. In Section 2, we discuss the problem of translation selection using collocation in MT. In Section 3, we describe the basic concepts of WordNet and k-NN classification and their use in translation selection. In Section 4, we present the method for optimizing the collocation dictionary for efficient selection of translations. In Section 5, we describe experimental results and a brief comparative study with other semantic clustering methods like Latent Semantic Analysis (Landauer et al., 1998; Shin, 1999). Section 6 comprises the conclusion and further work.

2. Collocation-Based Selection of Transitive Verbs

Verbs play a key role in the whole semantics of a sentence. Therefore, a good translation of verbs is essential for improving the accuracy of the translated sentence. Hence, for accurate determination of target verbs, collocation information is usually used.

For example, Dagan and Itai (1994) proposed a statistical model for word-sense disambiguation, using collocation information in the context of Hebrew–English MT. For a syntactic tuple, such as subject–verb and verb–object, in the source language, they collected the alternative target syntactic tuples with their counts in the target-language corpus. Then, the target word is selected as the one with the highest frequency.

Kim and Kim (1994) used a collocation-based method for the disambiguation of target words in Korean–English MT. Here, verb–object pairs are extracted from a Korean corpus to build a collocation dictionary. The dictionary is then used to select the target verb in English.

Similar to Kim and Kim's approach, we use a collocation dictionary for selecting target words. The structure of the dictionary is as in (1) (Kim and Kim, 1998),

$$T(S_i) = \begin{cases} T_1 & \text{if } \mathrm{Cooc}(S_i, S_1) \\ T_2 & \text{if } \mathrm{Cooc}(S_i, S_2) \\ & \dots \\ T_n & \text{otherwise,} \end{cases} \tag{1}$$

where $\mathrm{Cooc}(S_i, S_j)$ denotes the co-occurrence of source words S_i and S_j and T_j is the target word. $T(\cdot)$ denotes the translation process.

Table I shows a collocation dictionary for English verb S_i = *build*. The dictionary shows that the word *build* is translated into five different target words in Korean, depending on the context. For example, *build* is translated into *geon-seol-ha-da* ('construct') when its object noun is a noun *plant* ('factory'), into *che-chak-ha-da* ('produce') when co-occurring with the object noun *car*, and into *seol-lip-ha-da* ('establish') in the context of object noun *company* (Table II).

One of the fundamental difficulties in collocation-based approaches to word-sense disambiguation (target word selection in our case) is the problem of data sparseness or unseen words. For example, for an unregistered object noun like *vehicle*, the correct translation of the verb cannot be selected using the dictionary described above. In the next section, we present a method that resolves this problem by using WordNet and k-NN learning.

Table I. Examples of collocations for a verb *build* in the dictionary

Meaning of *build* in Korean (T_j)	Collocated object noun (S_j)			
geon-seol-ha-da (= 'construct')	plant	facility	network	...
geon-chook-ha-da (= 'design')	house	center	housing	...
che-chak-ha-da (= 'produce')	car	ship	model	...
seol-lip-ha-da (= 'establish')	company	market	empire	...
koo-chook-ha-da (= 'develop')	system	stake	relationship	...

Table II. Examples of translation of *build*

Source words		Translated words (in Korean)	Sense of the verb
build a *plant*	⇒	*gong-jang*-eul geon-seol-ha-da	'construct'
build a *car*	⇒	*ja-dong-cha*-reul che-chak-ha-da	'produce'
build a *company*	⇒	*hoi-sa*-reul seol-lip-ha-da	'establish'

3. WordNet and *k*-NN Learning

The translation for an unregistered object noun in the collocation dictionary is selected by using *k*-NN learning on WordNet. In this section, we define the semantic distance in WordNet and also describe how the translation of verbs is accomplished by a *k*-NN search.

3.1. SEMANTIC DISTANCE IN WORDNET

WordNet (Miller et al., 1990; Richardson et al., 1994; Fellbaum, 1998) represents the lexical knowledge of a native speaker of English as a tree-like structure (Figure 1).

The basic semantic relation in WordNet is *synonymy*. Sets of synonyms are called "synsets" and form the basic building blocks. A lexical concept is represented in WordNet by the set of synonyms that can be used to express that concept. Although synonymy is a semantic relation between word forms, the semantic relation that is most important in organizing WordNet is a relation between lexical concepts. It is the relation of subordination (or class inclusion or subsumption), which is called "hyponymy".

In Figure 1, the word *boy* is a synonym for *child* and the two words form a synset 1.1.1.1.31.2. This synset is in hyponym relation to the synset 1.1.1.1.31, *male*. In the same manner, the words *entity* and *human* are "hypernyms" (superordinates) of the word *male*, and *boy* is a hyponym of that word.

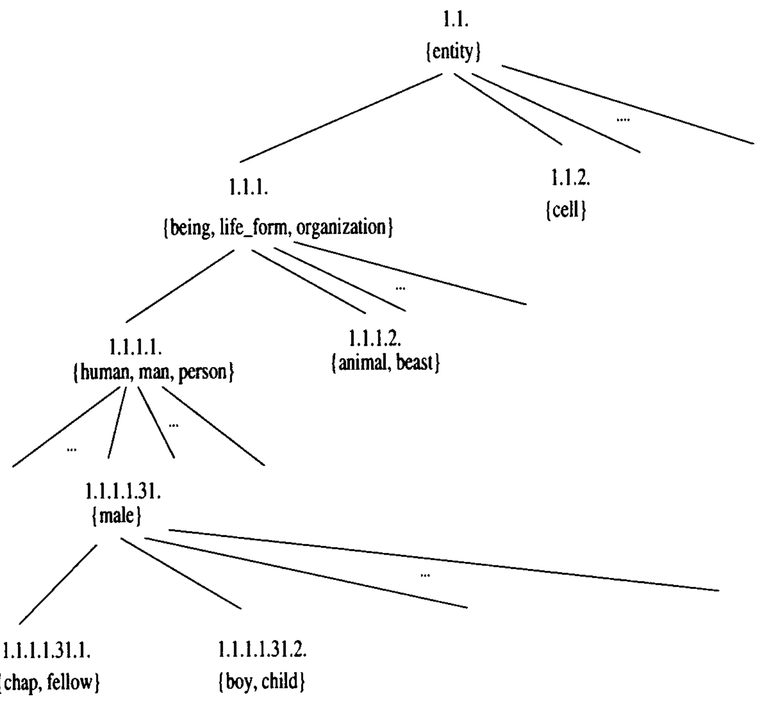

Figure 1. A partial structure for the node {1.1.} (*entity*) on WordNet.

A synset in WordNet represents the semantic concept of a word and the semantic distance between two words is defined as the distance between the two synsets in WordNet.

For the calculation of the semantic distance, each synset in WordNet is assigned an M-value. We define the M-value as (2),

$$M\text{-value} = \frac{radix}{sf^{p}} \tag{2}$$

where *radix* is an initial M-value, *sf* is a scale factor, and p is the number of edges from the root to the synset. WordNet branches out more minutely in its lower part than its upper ones. A link in a lower WordNet indicates a shorter semantic distance than a link in higher WordNet in our opinion. Figure 2 shows examples of M-values with the *radix* value of 8.0 and *sf* value of 2.0.

We define the distance between two synsets on a route as the difference between their M-values. For example, the distance between 4.1 and 4.1.4.8.1 can be calculated as $2.0 - 0.25 = 1.75$. Similarly, the distance between any two synsets that are not on a route is defined as the sum of the two distances between the “comset” and the two synsets. A comset is defined as the most specific synset that is superordinate to both synsets. In Figure 2, 4.1.4.8 is the comset of 4.1.4.8.1.10 and 4.1.4.8.2.1.

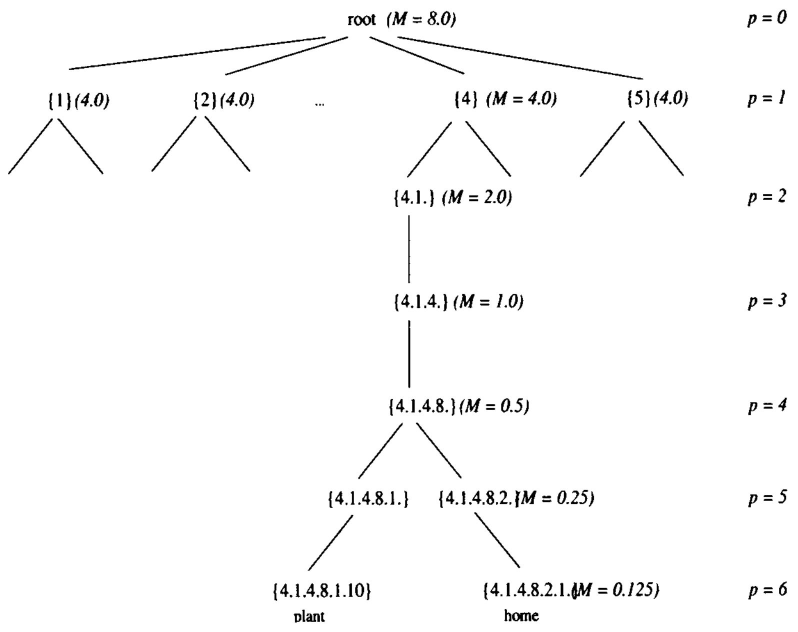

Figure 2. WordNet with the M-values.

The procedure for computing the semantic distance from the noun *plant*(4.1.4.8.1.10) to the noun *home*(4.1.4.8.2.1) which are not on a route can be explained as follows:

1. Find the comset (4.1.4.8) for the synsets *plant* and *home*.
2. Calculate the M-values of the comset and each synset.

$$\text{comset} : \frac{8.0}{2.4^4} = 0.5$$

$$\textit{plant}, \textit{home} : \frac{8.0}{2.0^7} = 0.125$$

3. Calculate the distances $0.5 - 0.125 = 0.375$ for each from the comset to the synsets.
4. Add the two distances for the distance between the two synsets.

So the distance between synset *plant* and synset *home* is simply as in (3).

$$\begin{aligned} D(\textit{plant}, \textit{home}) &= 0.375 + 0.375 \\ &= 0.75 \end{aligned} \tag{3}$$

Once the M-values are assigned to all synsets, the semantic distance can be calculated efficiently.

3.2. k-NN LEARNING FOR TRANSLATION SELECTION

The semantic distance in WordNet is required when performing a k-NN search to select the translation of a verb. The nearest instance of a given word is decided by using the semantic distance as the distance between the two instances.

The k-NN learning algorithm (Cover and Hart, 1967; Aha et al., 1991; Mitchell, 1997) assumes all instances correspond to points in the n-dimensional space R^n. The nearest neighbors of an instance are defined in terms of the standard Euclidean distance.

Then the distance between two instances x_i and x_j, $D(x_i, x_j)$, is defined as in (4),

$$D(x_i, x_j) = \sqrt{(a(x_i) - a(x_j))^2} \tag{4}$$

and $a(x_i)$ denotes the value of instance x_i. Let us consider learning discrete-valued target functions of the form f: $R^n \rightarrow V$, where V is the finite set $\{v_1, \ldots, v_s\}$. The k-NN algorithm for approximating a discrete-valued target function is given in Figure 3.

The value $\hat{f}(x_1)$ returned by this algorithm as its estimate of $f(x_q)$ is just the most common value of f among the k training examples nearest to x_q. If we choose $k = 1$, then the 1-NN algorithm assigns to $\hat{f}(x_q)$ the value $f(x_i)$ where x_i is the training instance nearest to x_q. We use $k = 1$ for the translation selection in this paper.

The following is an example of translation selection of the verb *build* for the verb–object pair *build–highway* using WordNet and the k-NN search.

1. For the object *highway*, obtain the sense 1.1.4.3.19.5.6 from WordNet.
2. To estimate the translation of the verb, find the nearest training example x_i to the query example x_q.
   For x_q = 1.1.4.3.19.5.6 (*highway*), we get the nearest example x_i = 1.1.4.3.19.5 *road*) from the dictionary.

– Training
  - For each training example $\langle x, f(x) \rangle$, add the example to the list *training_examples*.

– Classification
  - Given a query instance x_q to be classified,
    * Let $x_1, \ldots, x_k$ denote the k instances from *training_examples* that are nearest to x_q.
    * Return

$$\hat{f}x_q) \leftarrow \arg\max_{v \in V} \sum_{i=1}^{k} \delta(v, fx_i)) \tag{5}$$

where $\delta(a, b) = 1$ if $a = b$ and $\delta(a, b) = 0$ otherwise.

Figure 3. The k-NN learning algorithm.

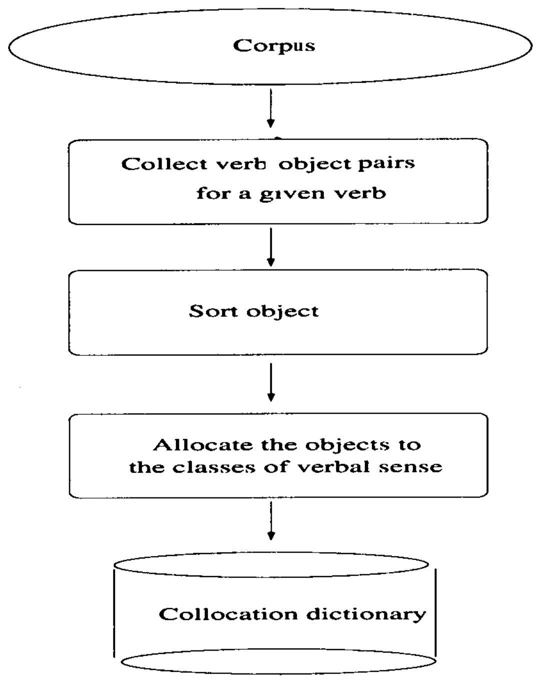
Corpus
Collect verb object pairs for a given verb
Sort object
Allocate the objects to the classes of verbal sense
Collocation dictionary

Figure 4. The procedure for collocation dictionary construction.

3. So the target function $\hat{f}(1.1.4.3.19.5.6)$ comes from $f(1.1.4.3.19.5)$, and *geon-seol-ha-da* ('construct') in Korean is decided as the translation of the verb for the object noun *highway*, which is the correct Korean word.

4. Optimization of the Collocation Dictionary

A collocation dictionary should be built for housing the collocation information for target-word selection. The dictionary size can be reduced for efficiency because of the use of k-NN search based on semantic distance in WordNet. In this section, we describe how to construct the collocation dictionary, and how to optimise the size of the dictionary.

4.1. CONSTRUCTION OF A COLLOCATION DICTIONARY

The procedure for constructing a collocation dictionary is as shown in Figure 4.

1. Collect all verb–object pairs for a given verb from the corpus. The sentences are analyzed syntactically with a parser, and the pairs are collected
2. Sort the object nouns into a list in descending order of frequency.
3. Set up the classes for translation of the verb in the target language. For a given verb, the possible target words are determined by using a bilingual dictionary and the former translated results.

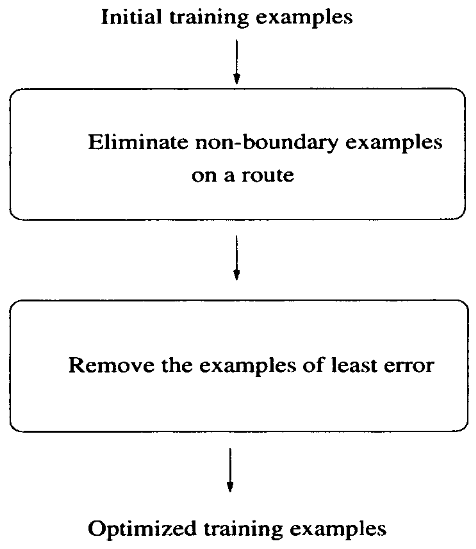
Initial training examples
Eliminate non-boundary examples on a route
Remove the examples of least error
Optimized training examples

Figure 5. The procedure for dictionary optimization.

4. Allocate the object nouns from the sorted list to the respective classes of the verb translations in the order of frequency. In this step, the translations are manually determined by the context of the sentence.
5. Repeat 4 until the end of the list.

As a result, in the collocation dictionary as shown in Section 2, each class of verb translation will have a list of object nouns for training examples in descending order of frequency.

4.2. OPTIMIZATION OF A COLLOCATION DICTIONARY

The collocation dictionary constructed by the above procedure shown in Figure 3 can be optimized by the use of k-NN. The optimization process is performed in two steps (Figure 5). First, the *non-boundary* examples on routes that never affect the translation accuracy are eliminated. Then the examples of least error are removed. The optimized training examples are then converted into the optimized collocation dictionary.

4.2.1. *Elimination of non-boundary examples on a route*

This step tries to find "non-boundary" examples on a route (which starts from the root node and ends at a leaf node of WordNet) and eliminates the non-boundary examples to reduce the size of the collocation dictionary. We define the "boundary" examples as the two neighbor examples which are close to each other, and do not belong to the same class of verbal translation. For the definition of classes, a bilingual dictionary and a target-word corpus are needed. Several human experts

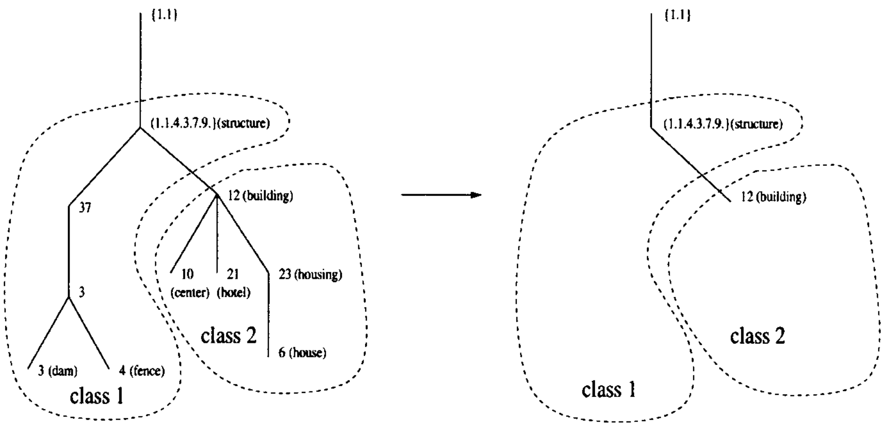

Figure 6. Elimination of non-boundary examples.

in MT are involved in defining the class. The boundary examples can include the usage of non-boundary examples on the same route without losing any accuracy for translation selection.

The procedure for eliminating non-boundary examples is as follows:

1. All training examples with their M-values are assumed to be mapped to nodes of WordNet.
2. Find all boundary examples on a route, and eliminate all non-boundary examples on the same route.
3. Add the frequency of each non-boundary example to the frequency of the nearest boundary example.

For example, in Figure 6(a), two classes are shown on WordNet with their synset relations. Synset 1.1.4.3.7.9 and synset 1.1.4.3.7.9.1 are boundary examples, because they are the nearest to each other on the same route but do not belong to the same class. The non-boundary examples are all eliminated as shown in Figure 6(b) for the routes. The boundary synsets remain in the collocation dictionary and the frequencies of the eliminated examples such as *center*, *hotel*, *house*, and *housing* are all added to the frequency of the nearest boundary example *building*. The frequency of each synset will be used for reduction of dictionary size; less frequently appearing data is first eliminated in the dictionary reduction process.

4.2.2. *Removing examples of least error*

After step 1, to reduce training examples, the examples of least error that do not much affect the correctness of the translation are removed from the training example set. To do this, we define the degree of similarity among the verb translation as in Figure 7.

The procedure for removing the examples of least error using the degree of similarity is as follows:

1. Construct the collocation matrix $CM = (c_{kn})$, where k is the index for the class of verb translations and n is training examples in each class.
2. The degree of similarity matrix among the verb translations is $SM = (s_{ij})$, where i, j are indices of the verb translations. s_{ij} is defined as

$$s_{ij} = \frac{\sum_{m=1}^{l} \sum_{n=1}^{l'} D(c_{im}, c_{jn})}{l \times l'} \tag{6}$$

where l and l' are the length of the list in each class, respectively and $D(c_{im}, c_{jn})$ is a distance measure defined in Equation (4). s_{ij} satisfies $s_{ij} = s_{ji}$ and $s_{ii} = 0$.

Figure 7. Procedure for computing the degree of similarity among verb translations.

1. Compute the degree of similarity among the translations of the verb using the procedure described in Figure 7.
2. To find the training example of least error from the current training example set,
   (a) Remove one training example n from the current training example set.
   (b) Compute the sum of the degrees of similarity of all test examples. For test example x_t, the degree of similarity $s_{\hat{f}(x_t)f(x_t)}$ between $\hat{f}(x_t)$ and $f(x_t)$ is computed.
   (c) Calculate the error E_n for the nth training example (see Equation (7) below).
   (d) Restore the training example set by returning the example n to the current example set.
   (e) Repeat (a)–(d) for all other training examples.
   (f) Find the example of least error.
3. Remove the example of least error.
4. Repeat 2–3 until not necessary.

The number of training examples is reduced while this procedure continues. In Step 2c, the error E_n for the nth training example is computed as in (7).

$$\begin{aligned} E_n &= S_n \times F_n \\ &= \sum_{x_t \in T} s_{\hat{f}(x_t)f(x_t)} \times \left(1 + \frac{f_n}{f_{\max}}\right) \end{aligned} \tag{7}$$

Here, $S_n = \sum_{x_t \in T} s_{\hat{f}(x_t)f(x_t)}$ is the sum of the degrees of similarity among the verb translations $\hat{f}(x_t)$ and $f(x_t)$, where $x_t \in T$ and T is the test example set. The term $F_n = (1 + \frac{f_n}{f_{\max}})$ is the weight corresponding to the frequency of the nth training example and has a value between 1 and 2. $f_{\max}$ denotes the maximum frequency of the examples in the training example set.

Table III. The number of training and test examples in 5-fold cross validation for *build*. Shown in parentheses are the numbers of sentences in which the training examples appear. For test examples, the numbers of unregistered words are also shown in the last row.

	Run	1	2	3	4	5	Ave.
Training	# examples	241	248	240	241	246	243.2
sample	(# sentences)	(726)	(726)	(721)	(727)	(727)	(726.4)
Test	# examples	182	182	182	181	181	181.6
sample	# unregistered	21	15	16	15	23	18

The error E_n is calculated by multiplying the degree of error S_n that represents an error without the nth example, and factor F_n of the nth example in the training data. The example that causes the least error and rarely appears in the corpus is preferentially removed from the dictionary. A degree of error S_n is computed by summing the degree of similarity among the same translation $f(x_t)$, and the translation result $\hat{f}(x_t)$ of test data x_t. The above procedure removes examples with low effect on translation. If S_n and F_n for each example in training data were already stored, performance of this procedure can be increased.

The above algorithm runs while the value of E_n is within a permitted limit.

4.3. EXPERIMENTAL EVALUATION

The effect of optimizing the collocation dictionary was investigated in terms of translation accuracy and time efficiency. The first set of experiments was performed on the transitive verb *build*. We extracted 908 example sentences containing *build*–object pairs from the *Wall Street Journal* (220,047 sentences in total) and other newspapers (41,750 sentences in total). The performance was measured using 5-fold cross-validation (Breiman and Spector, 1992; Cherkassky and Mulier, 1998: 78–80). A total of 908 sentences was divided into five disjoint samples and each sample became a test sample once in the experiment and the remaining four samples together become the training sample. An actual training set was then constructed from the training sentences by keeping only one instance for a verb–object pair. Table III shows the structure of the training and test sets for 5-fold cross validation. The number of unregistered words is also shown for the test sets. Appendix A gives a formal description of the cross-validation procedure.

4.4. EXPERIMENTS WITH *build*

Table IV shows the effects of dictionary size optimization on translation accuracy and translation time for *build*. Compared are the cases of baseline algorithm (no optimization of dictionary size and no k-NN learning), the case for using only k-NN

Table IV. Translation accuracy, dictionary size, and translation time for *build*. The first row shows the performance for the baseline algorithm in which the dictionary is not optimized. The second row is the performance for the case when k-NN is used but the dictionary is not optimized. The following rows show the performance when dictionary optimization is attempted and k-NN search is used to selection the best translation. In case of dictionary optimization, the performances after the first optimization step and the second step are also compared. The relative accuracy and time are the performances of each method compared against those without k-NN learning. The number in parentheses in the third column denotes the percentage of the remaining collocation entries after the first optimization step. "Hit ratio" denotes the percentage of direct matches in the remaining dictionary.

	Optimi-zation	Dictionary size (%)	Accuracy (%)	Rel. accuracy	Time (sec)	Rel. time	Hit ratio
Baseline	None	100	74.44	1.00	0.026	1.00	74.44
k-NN	None	100	87.68	1.18	0.096	3.69	74.44
	1st opt.	54.52	87.68	1.18	0.122	4.69	41.48
		49.07 (90)	88.56	1.19	0.114	4.38	40.44
		38.16 (70)	88.58	1.19	0.104	4.00	38.26
	2nd opt.	27.26 (50)	87.52	1.18	0.090	3.46	35.44
		16.36 (30)	86.28	1.16	0.080	3.08	30.56

learning without dictionary optimization, and the cases for dictionary optimization using k-NN learning.

The accuracy of translation is measured in two different ways. When k-NN is not used, the error count $e_i(x)$ increases if the query word x is not found in the dictionary, i.e.,

$$e_i(x) = \begin{cases} 0 & \text{if found,} \\ 1 & \text{otherwise} \end{cases} \tag{8}$$

When using k-NN search, each test example x searches its k nearest examples $f_i(x)$ in the training set, and increments its error count $e_i(x)$ if an incorrect example is found, i.e.,

$$e_i(x) = \begin{cases} 0 & \text{if } f_i(x) = y, \\ 1 & \text{otherwise} \end{cases} \tag{9}$$

The results can be summarized as follows. When the dictionary was not optimized, the average accuracy of translation selection was 74.44% without k-NN search. When k-NN was used, the performance suggests the importance of k-NN search for the determination of target words. However, using k-NN, more translation time was required (0.026 sec. without k-NN and 0.096 sec. with k-NN). This is due to the increased time for k-NN search steps. This result motivated us to find a method for optimizing the dictionary size that further reduces translation time, without deteriorating translation accuracy. This is why multiple optimization steps are used.

The results of the first optimization step are shown in Table IV. Comparing the performances before and after the first optimization step, the average accuracy is the same (87.68%), but the first optimization step has incurred an additional translation time (from 0.096 sec. before the first optimization to 0.122 sec. after the optimization). This is because the additional optimization has reduced the number of dictionary entries and thus k-NN search was used more frequently.

For further reduction of translation time, we used the degree of similarity among verb translations during the second optimization step. Table IV shows the effect. It is interesting to note that the translation accuracy was increased for a short while by the second optimization step from 87.68% to 88.58%, while dictionary size was reduced to 38.16% of its initial size. This seems due to the noisy examples in the collocation dictionary that were eliminated during the second optimization step.

During the optimization phase, the hit ratio decreased steadily from 74.44% to 30.56%, whereas the translation accuracy varied slightly between 87.68% and 86.28%. This fact suggests that many of the examples in the dictionary do not affect the translation accuracy significantly.

4.5. EXPERIMENTS ON CORE TRANSLATIONS

The translation accuracy can be improved further by maintaining the default target words, i.e., *core translation*, of verbs for the object nouns unregistered in WordNet. Table V shows the translation accuracy when core translation was not used. Translation accuracy improved by 7.4% on average when the unregistered examples were allocated to the class of its core translation. Table VI summarises the translation accuracy when the core translations were used for verbs *build* and five other verbs. The five other verbs had most frequently appeared in the corpus, and at the same time, they have definitely classified target words in English–Korean MT.

Figure 8 shows the change in translation accuracy as a function of dictionary size for *build* and other verbs. The two curves in the figure are different in the trend showing translation accuracy, but quite similar in the trend of accuracy change as a function of dictionary size. They do not change much until the size is reduced to 40% of the initial dictionary size, and then change rapidly from the 20% of the initial size. From this, we may infer that more than half of the examples are not essential for the translation selection. The curves for *build* and five other verbs have invariant regions around their starting points. This is because the first optimization step did not affect translation accuracy.

Figure 9 shows the translation time for the verbs in the study. Actually, translation time is composed of dictionary lookup time and other operations. However, most of the translation time is lookup time. In the figure, two lines are shown,

Table V. Translation accuracy for *build* and other verbs when core translation was not used. The results show how accuracy changes as the dictionary size is reduced. The reduction of dictionary size in percent is computed by dividing the number of remaining collocation entries by the number of initial collocation entries.

Verb	Dictionary size (%)					
	100	90	70	50	30	0
build	87.7	87.7	87.7	88.4	88.1	30.4
Other verbs	82.1	82.3	82.1	81.8	79.8	65.3

Table VI. Translation accuracy and translation time for *build*, and five other verbs when core translation was used. The results show the change in accuracy and time as the dictionary size is reduced.

Verb	Experiment	Dictionary size (%)					
		100	90	70	50	30	0
build	Acc. (%)	91.0	91.0	91.0	91.7	91.4	33.7
	Time (sec.)	0.096	0.102	0.118	0.116	0.094	0
Other verbs	Acc. (%)	90.3	90.5	90.3	88.0	89.7	72.5
	Time sec.)	0.102	0.096	0.096	0.087	0.072	0

an upper line for *build* and a lower line for the five other verbs. The difference between the two lines is caused from the size of examples. *Build* has over 900 examples while other verbs have only approximately from 150 to 500 examples in our corpus. We extracted *build*–object pairs from several corpora initially. However, when extracting other verb–object pairs, we used relatively small corpora for testing. In spite of the difference between the two lines, the trend is very similar.

4.6. A COMPARATIVE STUDY WITH LATENT SEMANTIC ANALYSIS

Latent Semantic Analysis (LSA) is a theory and method for extracting and representing the contextual-usage meaning of words, by statistical computations applied to a large corpus of text. LSA is composed of two steps. First, the text is represented as a matrix in which each row stands for a unique word, and each column stands for a text passage or other context. Second, LSA applies singular value decomposition (SVD) to the matrix and generates three matrices; one describes original row entities as vectors of derived orthogonal factor value, the second describes the original column entities in the same way, and the third is a diagonal matrix containing scaling values (Landauer et al., 1998). In the case of the original

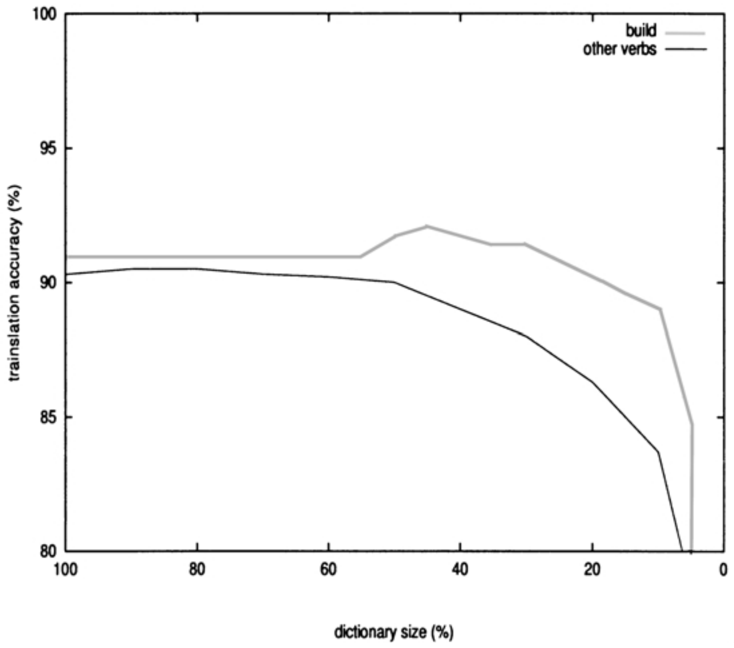

Figure 8. The size of examples and translation accuracy.

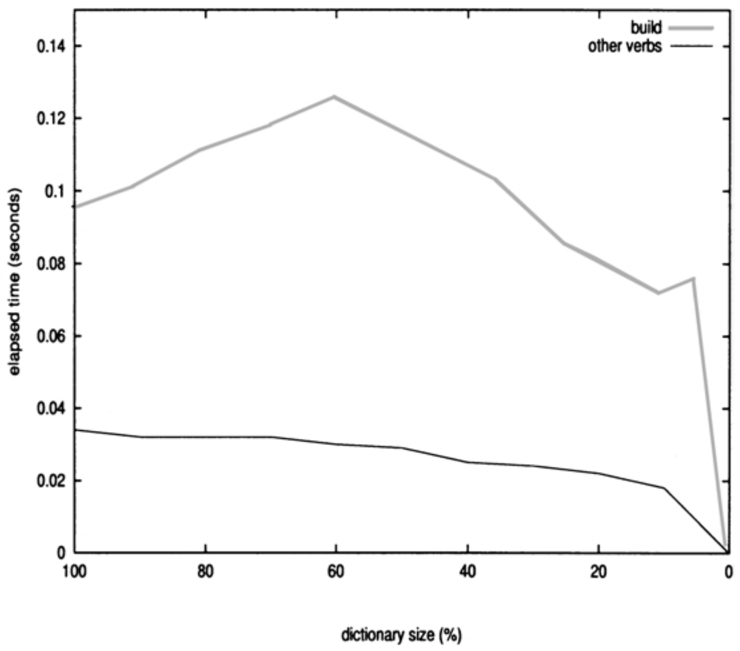

Figure 9. The size of examples and the translation time.

Table VII. Similar word list extracted by WordNet and LSA. The first column shows words to be used to find their own alternative words. The second column shows similar words found from WordNet, and the third column shows the same lists from LSA. An empty cell means that the word in the first column could not be found in the database implemented with LSA methodology.

Word	Similar words	
	WordNet	LSA
house	duplex, hacienda, cabin	chamber, commons, president
plant	distillery, factory, manufactory	
facility	arborectum, deposit, depository	
network	intercom, radio, wireless	resau, programming, Ontario
center	building, edifice, chancellery	
housing	lodging, apartment, flat	
car	auto, automobile, machine	remark, matter, example
ship	derelict, pirate, steamship	
company	service, utility, raider	enterprise, material, corporation
market	class, craft, trade	subvention, gaz(= gazette), refinery
empire	conglomerate, business, concern	
stake	bet, wager, ante	looking, fine, attendant

document–term matrix, the document–vector matrix, term–vector matrix, and diagonal matrix are automatically generated. We are mainly interested in the second matrix, the term–vector matrix, since we wish to determine the similarity between two terms.

To compare LSA to the WordNet synsets, we decided to see whether it produced words similar to those in close neighborhoods in WordNet. A similar word plays the role of an alternative word for an unseen word, during translation. We used Telcordia Technologies' LSA search system, to find the most similar word. Table VII shows term–term similarity measured by WordNet and LSA.

Unfortunately, we faced several problems in applying LSA to our research. First, words that are not found in the database appear frequently. In our experiments, we extracted about 25,000 words that appear more than twice, and only 2,500 words with 2,500-more info-gain (McCallum, 1996) from a 3 million-word corpus. This causes another data sparseness problem. Second, similar words from LSA are not proper when used as alternative words in MT. The selectional restriction of a verb depends more on semantic type like human-being, place, etc. of an object noun, than on the context in a document. Translation of a verb was decided by comparing its object noun in a sentence. Collocated object nouns shown in Table I are likely to form a semantic type, not to be seen in a document. *Car* and *ship* are in the same semantic type as *vehicle*, but unlikely to co-occur in a document. Generally, *doctor* is more similar to *hospital* than *lawyer* when using contextual similarity like LSA (Karov and Edelman, 1998). However, when selecting a target word of *fire*, *hae-go-ha-da* (= ‘lay off’) is selected for the object noun *doctor* and *lawyer* being a human-being type and *sso-da* (= ‘shoot’) is selected for *gun/rifle* being a weapon type.

5. Conclusion

We have presented a method for optimizing collocation dictionaries to select accurate translations of verbs. Using WordNet and k-NN search, we could eliminate approximately half of the dictionary entries without loss of translation accuracy.

The optimization process proceeds in two steps. The first step involves eliminating non-boundary examples in WordNet which do not affect the translation accuracy at all. The second optimization step eliminates the examples with least error. The dictionary entry for the verb *build* is still available with the reduction of training examples up to 60%, keeping translation accuracy level of 87%. We found dictionary optimization has the effect in eliminating noisy examples. The use of core translations, i.e., the default translations of target words, has further improved the translation accuracy. The dictionary lookup, translation time increased as the dictionary size was reduced to 60% of the initial size, due to the overhead for k-NN search.

For further improvement in the translation accuracy, some problems remain unsolved. First, WordNet itself misses some words and concepts. To compensate for this, other complementary algorithms using corpora need to be developed. Second, for more robust measurement of semantic distance, other logical operations on WordNet should be considered. We implemented two logical operations of WordNet, Hypernym and Hyponym. The semantic distance calculation will be more elaborate when other operations like Antonym, Meronym and Lolonym will be integrated with the established calculation.

In this paper, we mainly discussed the methodology to optimise the dictionary, not to construct the dictionary. The methodology for automatically constructing collocation dictionaries for MT should also be studied in the future. Several authors have proposed methods for automatic dictionary construction (Dorr et al., 1995; Riloff, 1996; Soderland et al., 1995; Takenobu et al., 1995). However, they have difficulties in reflecting the linguistic customs of target language speakers. A verb such as *build* has a single meaning in an English dictionary; ‘make something by joining things together’ (Sinclair, 1997). However, when it is translated into Korean, five target words are needed as shown in this paper. This results from differences in linguistic customs between English and Korean speakers. For building practical translation dictionaries, methods for learning the linguistic customs should be developed.

Last but not least, the optimization method proposed in this paper can be used to improve also the accuracy and efficiency of selecting translations for other words, including transitive verbs, intransitive verbs, modifiers, and prepositions.

Appendix A. *k*-Fold Cross-Validation

In the algorithm shown below, the average empirical error is measured for the examples of size n,

$$\mathbf{Z} = [\mathbf{X}, \mathbf{y}]$$

where $\mathbf{X} = [x_1, \ldots, x_n]$ and $\mathbf{y} = [y_1, \ldots, y_n]$. Here, x_i is the input example and y_i is the output for x_i.

1. Divide the examples $\mathbf{Z}$ into k disjoint samples of equal size, $\mathbf{Z}_1, \mathbf{Z}_2, \ldots, \mathbf{Z}_k$.
2. For each validation sample $\mathbf{Z}_i$ of size n/k,
   (a) Use the remaining examples, $\mathbf{Z}_i' = \cup_{j \neq i} \mathbf{Z}_j$ as the training example set.
   (b) For the examples in $\mathbf{Z}_i$, compute empirical error r_i as follows:

$$r_i = \frac{k}{n} \sum_{x \in \mathbf{Z}_i} e_i(x)$$

   where

$$e_i(x) = \begin{cases} 0 & \text{if } f_i(x) = y \\ 1 & \text{otherwise} \end{cases}$$

3. Compute the average error $AE(\mathbf{Z})$ by averaging the empirical error sums for $\mathbf{Z}_1, \mathbf{Z}_2, \ldots, \mathbf{Z}_k$.

$$AE(\mathbf{Z}) = \frac{1}{k} \sum_{i=1}^{k} r_i$$

References

Aha, David, Dennis Kibler, and Marc Albert: 1991, ‘Instance-Based Learning Algorithms’, *Machine Learning* **6**, 37–66.

Breiman Leo and Philip Spector: 1992, ‘Submodel Selection and Evaluation in Regression: The X-Random Case’, *International Statistic Review* **60**, 291–319.

Cherkassky, Vladimir and Filip Mulier: 1998, *Learning from Data: Concepts, Theory, and Methods*, John Wiley, New York.

Cover, Thomas M. and Peter E. Hart: 1967, ‘Nearest Neighbor Pattern Classification’, *IEEE Transactions on Information Theory* **13**, 21–27.

Dagan, Ido and Alon Itai: 1994, ‘Word Sense Disambiguation Using a Second Language Monolingual Corpus’, *Computational Linguistics* **20**, 563–596.

Dagan, Ido, Lillian Lee, and Fernando Pereira: 1997, ‘Similarity-Based Methods for Word Sense Disambiguation’, *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, Madrid, Spain, pp. 56–63.

Dagan, Ido, Lillian Lee and Fernando C. N. Fereira: 1999, ‘Similarity-Based Models of Word Cooccurrence Probabilities’, *Machine Learning* **34**, 43–69.

Dorr, Bonnie J., Joseph Garman, and Amy Weinberg: 1995, ‘From Syntactic Encodings to Thematic Roles: Building Lexical Entries for Interlingual MT’, *Machine Translation* **9**, 71–100.

Fellbaum, Chistiane: 1998, *WordNet: An Electronic Lexical Database*, The MIT Press, Cambridge, MA.

Karov, Yael and Shimon Edelman: 1998, ‘Similarity-Based Word Sense Disambiguation’, *Computational Linguistics* **24**, 41–59.

Kim, Nari and Yung Taek Kim: 1994, ‘Determining Target Expression Using Parameterized Collocations from Corpus in Korean–English Machine Translation’, *Pacific Rim International Conference on Artificial Intelligence*, Beijing, China, pp. 732–736.

Kim, Yuseop and Yung Taek Kim: 1998, ‘Semantic Implementation Based on Extended Idiom for English to Korean Machine Translation’, *The Asia-Pacific Association for Machine Translation Journal* **21**, 23–39.

Landauer, Thomas K., Peter W. Foltz, and Darrell Laham: 1998, ‘An Introduction to Latent Semantic Analysis’, *Discourse Process* **25**, 259–284.

McCallum, Andrew Kachites: 1996, ‘Bow: A Toolkit for Statistical Language Modeling, Text Retrieval, Classification and Clustering’, http://www.cs.cmu.edu/∼mccallum/bow.

Miller, George A., Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller: 1990, ‘Introduction to WordNet: An On-Line Lexical Database’, *International Journal of Lexicography* **3**, 235–244.

Mitchell, Tom M.: 1997, *Machine Learning*, McGraw-Hill, New York.

Resnik, Phillip: 1995, ‘Disambiguating Noun Groupings with Respect to WordNet Senses’, *Proceedings of the Third Workshop on Very Large Corpora*, Cambridge, MA, pp. 54–68.

Richardson Ray, Alan F. Smeaton, and John Murphy: 1994, ‘Using WordNet as a Knowledge Base for Measuring Semantic Similarity between Words’, Working Paper CA-1294, School of Computer Applications, Dublin City University.

Riloff, Ellen: 1996, ‘An Empirical Study of Automated Dictionary Construction for Information Extraction in Three Domains’, *Artificial Intelligence* **85**, 101–134.

SNU: 2001, ‘E-Tran 2001’, http://nlp.snu.ac.lcr/E-Tran2001 [in Korean].

Shin, Dong Ho: 1999, ‘A Study on Content-Based Information Retrieval System Using LSA’, MS Thesis, Seoul National University.

Sinclair, John (ed.): 1997, *Collins Cobuild English Dictionary*, Collins, London.

Soderland, Stephen, David Fisher, Jonathan Aseltine, and Wendy Lehnert: 1995, ‘CRYSTAL: Inducing a Conceptual Dictionary’, *The 1995 International Joint Conference for Artificial Intelligence*, Montreal, Canada, pp. 1314–1319.

Takenobu, Tokunaga, Iwayama Makoto, and Tanaka Hozumi: 1995, ‘Automatic Thesaurus Construction Based on Grammatical Relations’, *The 1995 International Joint Conference for Artificial Intelligence*, Montreal, Canada, pp. 1308–1313.

Yarowsky, David: 1992, ‘Word-Sense Disambiguation Using Statistical Models of Roget’s Categories Trained on Large Corpora’, *Proceedings of the fifteenth [sic] International Conference on Computational Linguistics, COLING-92*, Nantes, France, pp. 454–460.