Near-Synonym Choice in an Intelligent Thesaurus

Diana Inkpen
School of Information Technology and Engineering,
University of Ottawa
800 King Edward, Ottawa, ON, Canada, K1N 6N5
diana@site.uottawa.ca

Abstract

An intelligent thesaurus assists a writer with alternative choices of words and orders them by their suitability in the writing context. In this paper we focus on methods for automatically choosing near-synonyms by their semantic coherence with the context. Our statistical method uses the Web as a corpus to compute mutual information scores. Evaluation experiments show that this method performs better than a previous method on the same task. We also propose and evaluate two more methods, one that uses anti-collocations, and one that uses supervised learning. To asses the difficulty of the task, we present results obtained by human judges.

1 Introduction

When composing a text, a writer can access a thesaurus to retrieve words that are similar to a given target word, when there is a need to avoid repeating the same word, or when the word does not seem to be the best choice in the context.

Our intelligent thesaurus is an interactive application that presents the user with a list of alternative words (near-synonyms), and, unlike standard thesauri, it orders the choices by their suitability to the writing context. We investigate how the collocational properties of near-synonyms can help with choosing the best words. This problem is difficult because the near-synonyms have senses that are very close to each other, and therefore they occur in similar contexts; we need to capture the subtle differences specific to each near-synonym.

Our thesaurus brings up only alternatives that have the same part-of-speech with the target word. The choices could come from various inventories of near-synonyms or similar words, for example the Roget thesaurus (Roget, 1852), dictionaries of synonyms (Hayakawa, 1994), or clusters acquired from corpora (Lin, 1998).

In this paper we focus on the task of automatically selecting the best near-synonym that should be used in a particular context. The natural way to validate an algorithm for this task would be to ask human readers to evaluate the quality of the algorithm's output, but this kind of evaluation would be very laborious. Instead, we validate the algorithms by deleting selected words from sample sentences, to see whether the algorithms can restore the missing words. That is, we create a *lexical gap* and evaluate the ability of the algorithms to *fill the gap*. Two examples are presented in Figure 1. All the near-synonyms of the original word, including the word itself, become the choices in the solution set (see the figure for two examples of solution sets). The task is to automatically fill the gap with the best choice in the particular context. We present a method of scoring the choices. The highest scoring near-synonym will be chosen. In order to evaluate how well our method works we consider that the only correct solution is the original word. This will cause our evaluation scores to underestimate the performance, as more than one choice will sometimes be a perfect solution. Moreover, what we consider to be the best choice is the typical usage in the corpus, but it may vary from writer to writer. Nonetheless, it is a convenient way of producing test data.

The statistical method that we propose here is based on semantic coherence scores (based on mutual information) of each candidate with the words in the context. We explore how far such a method can go when using the Web as a corpus. We esti-

Sentence: This could be improved by more detailed consideration of the processes of .... propagation inherent in digitizing procedures.
Original near-synonym: error
Solution set: mistake, blooper, blunder, boner, contretemps, error, faux pas, goof, slip, solecism

Sentence: The day after this raid was the official start of operation strangle, an attempt to completely destroy the .... lines of communication.
Original near-synonym: enemy
Solution set: opponent, adversary, antagonist, competitor, enemy, foe, rival

Figure 1: Examples of sentences with a lexical gap, and candidate near-synonyms to fill the gap.

mate the counts by using the Waterloo MultiText System (Clarke and Terra, 2003b) with a corpus of about one terabyte of text collected by a Web crawler. We also propose a method that uses collocations and anti-collocations, and a supervised method that uses words and mutual information scores as featured for machine learning. To better asses the difficulty of the task we present results obtained by two human judges.

2 Related work

The idea of using the Web as a corpus of texts has been exploited by many researchers. Grefenstette (1999) used the Web for example-based machine translation; Kilgarriff (2001) investigated the type of noise in Web data; Mihalcea and Moldovan (1999) and Agirre and Martinez (2000) used it as an additional resource for word sense disambiguation; Resnik (1999) mined the Web for bilingual texts; Turney (2001) used Web frequency counts to compute information retrieval-based mutual-information scores. In a *Computational Linguistics* special issue on the Web as a corpus (Kilgarriff and Grefenstette, 2003), Keller and Lapata (2003) show that Web counts correlate well with counts collected from a balanced corpus: the size of the Web compensates for the noise in the data. In this paper we are using a very large corpus of Web pages to address a problem that has not been successfully solved before.

In fact, the only work that addresses exactly the same task is that of Edmonds (1997), as far as we are aware. Edmonds gives a solution based on a lexical co-occurrence network that included second-order co-occurrences. We use a much larger corpus and a simpler method, and we obtain much better results.

Our task has similarities to the word sense disambiguation task. Our near-synonyms have senses that are very close to each other. In Senseval, some of the fine-grained senses are also close to each other, so they might occur in similar contexts, while the coarse-grained senses are expected to occur in distinct contexts. In our case, the near-synonyms are different words to choose from, not the same word with different senses.

Turney *et al.* (2003) addressed the multiple-choice synonym problem: given a word, choose a synonym for that word, among a set of possible solutions. In this case the solutions contain one synonym and some other (unrelated) words. They achieves high performance by combining classifiers. Clarke and Terra (2003a) addressed the same problem as Turney *et al.*, using statistical associations measures computed with counts from the Waterloo terabyte corpus. In our case, all the possible solutions are synonyms of each other, and the task is to choose one that best matches the context: the sentence in which the original synonym is replaced with a gap. It is much harder to choose between words that are near-synonyms because the context features that differentiate a word from other words might be shared among the near-synonyms. Therefore we need features that catch more subtle differences.

3 A statistical method for near-synonym choice

Our method computes a score for each candidate near-synonym that could fill in the gap. The near-synonym with the highest score is the proposed solution. The score for each candidate reflects how well a near-synonym fits in with the context. It is based on the mutual information scores between a near-synonym and the content words in the context (we filter out the stopwords).

The **pointwise mutual information** (PMI) between two words x and y compares the probability of observing the two words together (their joint probability) to the probabilities of observing x and y independently (the probability of occurring together by chance) (Church and Hanks, 1991): $\mathrm{PMI}(x, y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$

The probabilities can be approximated by: $P(x) = C(x)/N$, $P(y) = C(y)/N$, $P(x, y) = C(x, y)/N$, where C denotes frequency counts and N is the total number of words in the corpus. Therefore: $\mathrm{PMI}(x, y) = \log_2 \frac{C(x,y) \cdot N}{C(x) \cdot C(y)}$, where N

can be ignored in comparisons, since is it the same in all the cases.

We model the context as a window of size $2k$ around the gap (the missing word): k words to the left and k words to the right of the gap. If the sentence is $s = \cdots w_1 \cdots w_k \ Gap \ w_{k+1} \cdots w_{2k} \cdots$, for each near-synonym NS_i from the group of candidates, the semantic coherence score is computed by the following formula:

$$Score(NS_i, s) = \Sigma_{j=1}^{k} \mathrm{PMI}(NS_i, w_j) + \Sigma_{j=k+1}^{2k} \mathrm{PMI}(NS_i, w_j).$$

We also experimented with the same formula when the sum is replaced with maximum to see whether a particular word in the context has higher influence than the sum of all contributions (though the sum worked better).

Because we are using the Waterloo terabyte corpus and we issue queries to its search engine, we have several possibilities of computing the frequency counts. $C(x, y)$ can be the number of co-occurrences of x and y when y immediately follows x, or the distance between x and y can be up to q. We call q the query frame size. The tool for accessing the corpus allows us to use various values for q in queries. We used queries of the type $[q] > (x..y)$, which asks how many times x is followed by y in a frame of size q.

The search engine also allows us to approximate words counts with document counts. If the counts $C(x)$, $C(y)$, and $C(x, y)$ are approximated as the number of document in which they appear, we obtain the PMI-IR formula (Turney, 2001). The queries we need to send to the search engine are the same but they are restricted to document counts: $C(x)$ is the number of document in which x occurs; $C(x, y)$ is the number of documents in which x is followed by y in a frame of size q.

Other statistical association measures, such as log-likelihood, could be used. We tried only PMI because it is easy to compute on a Web corpus and because (Clarke and Terra, 2003a) showed that PMI performed better than other measures.

We present the results in Section 6.1, where we compare our method to a baseline algorithm that always chooses the most frequent near-synonyms and to Edmonds's method for the same task, on the same data set. First, however, we present two other methods to which we compare our results.

ghastly mistake	spelling mistake
∗ghastly error	spelling error
ghastly blunder	∗spelling blunder
∗ghastly faux pas	∗spelling faux pas
∗ghastly blooper	∗spelling blooper
∗ghastly solecism	∗spelling solecism
∗ghastly goof	∗spelling goof
∗ghastly contretemps	∗spelling contretemps
∗ghastly boner	∗spelling boner
∗ghastly slip	∗spelling slip

Table 1: Examples of collocations and anti-collocations. The $*$ indicates the anti-collocations.

4 The anti-collocations method

For the task of near-synonym choice, another method that we implemented is the anti-collocations method. By *anti-collocation* we mean a combination of words that a native speaker would not use and therefore should not be used when automatically generating text. This method uses a knowledge-base of collocational behavior of near-synonyms acquired in previous work (Inkpen and Hirst, 2006). A fragment of the knowledge-base is presented in Table 1, for the near-synonyms of the word *error* and two collocate words *ghastly* and *spelling*. The lines marked by $*$ represent anti-collocations and the rest represent strong collocations.

The anti-collocations method simply ranks the strong collocations higher than the anti-collocations. In case of ties it chooses the most frequent near-synonym. In Section 6.2 we present the results of comparing this method to the method from the previous section.

5 A supervised learning method

We can also apply supervised learning techniques to our task. It is easy to obtain labeled training data, the same way we collected test data for the two unsupervised methods presented above. We train classifiers for each group of near-synonyms. The classes are the near-synonyms in the solution set. The word that produced the gap is the expected solution, the class label; this is a convenient way of producing training data, no need for manual annotation. Each sentence is converted into a vector of features to be used for training the supervised classifiers. We used two types of features. The features of the first type are the PMI scores of the left and right context with each class (each near-synonym from the group). The number of features of this type is twice the number of classes, one score for the part of the sentence at the left of the gap, and one for the part at the right of the gap. The features of the second type are the words in the context window. For each group of near-synonyms, we used as features the 500 most-frequent words situated close to the gaps in a development set. The value of a word feature for each training example is 1 if the word is present in the sentence (at the left or at the right of the gap), and 0 otherwise. We trained classifiers using several machine learning algorithms, to see which one is best at discriminating among the near-synonyms. In Section 6.3, we present the results of several classifiers.

A disadvantage of the supervised method is that it requires training for each group of near-synonyms. Additional training would be required whenever we add more near-synonyms to our knowledge-base. An advantage of this method is that we could improve the accuracy by using a combination of classifiers and by trying other possible features. We think that part-of-speech features of the content words in the context may not be very useful since all the possible solutions have the same part-of-speech and might have similar syntactic behavior. Maybe some function words immediately before the gaps could discriminate among the near-synonyms in some groups.

6 Evaluation

6.1 Comparison to Edmonds's method

In this section we present results of the statistical method explained in Section 3. We compare our results with those of Edmonds's (1997), whose solution used the texts from the year 1989 of the Wall Street Journal (WSJ) to build a lexical co-occurrence network for each of the seven groups of near-synonyms from Table 2. The network included second-order co-occurrences. Edmonds used the WSJ 1987 texts for testing, and reported accuracies only a little higher than the baseline. The near-synonyms in the seven groups were chosen to have low polysemy. This means that some sentences with wrong senses of near-synonyms might be in the automatically produced test set, but hopefully not many.

For comparison purposes, in this section we use the same test data (WSJ 1987) and the same groups of near-synonyms (we call these sentences the Exp1 data set). Our method is based on mutual information, not on co-occurrence counts. Our counts are collected from a much larger corpus. If we would have used groups of synonyms from WordNet, we would probably obtain similar results, because the words in seven groups differ very little.

1. mistake, error, fault
2. job, task, chore
3. duty, responsibility, obligation
4. difficult, hard
5. material, stuff
6. put up, provide, offer
7. decide, settle, resolve, adjudicate.

Table 2: The near-synonym groups used in the Exp1 data set.

		Accuracy			
Set	No. of cases	Base-line	Edmonds method	Stat. method (Docs)	Stat. method (Words)
1.	6,630	41.7%	47.9%	**61.0**%	59.1%
2.	1,052	30.9%	48.9%	**66.4**%	61.5%
3.	5,506	70.2%	68.9%	69.7%	**73.3**%
4.	3,115	38.0%	45.3%	64.1%	**66.0**%
5.	1,715	59.5%	64.6%	68.6%	**72.2**%
6.	11,504	36.7%	48.6%	52.0%	**52.7**%
7.	1,594	37.0%	65.9%	74.5%	**76.9%**
AVG	31,116	44.8%	55.7%	65.1%	**66.0%**

Table 3: Comparison between the statistical method from Section 3, baseline algorithm, and Edmonds's method (Exp1 data set).

Table 3 presents the comparative results for the seven groups of near-synonyms (we did not repeat them in the first column of the table, only the number of the group.). The last row averages the accuracies for all the test sentences. The second column shows how many test sentences we collected for each near-synonym group. The third column is for the baseline algorithm that always chooses the most frequent near-synonym. The fourth column presents the results reported in (Edmonds, 1997). column show the results of the supervised learning classifier described in Section 5. The fifth column presents the result of our method when using document counts in PMI-IR, and the last column is for the same method when using word counts in PMI. We show in bold the best accuracy for each data set. We notice that the automatic choice is more difficult for some near-synonym groups than for the others. In this paper, by accuracy we mean the number of correct choices made by each method (the number of gaps that were correctly filled). The correct choice is the near-synonym that was initially replaced by the gap in the test sentence.

To fine-tune our statistical method, we used the data set for the near-synonyms of the word *difficult* collected from the WSJ 1989 corpus as a development set. We varied the context window size k and the query frame q, and determined good values for the parameter k and q. The best results were obtained for small window sizes, $k = 1$ and $k = 2$ (meaning k words to the left and k words to the right of the gap). For each k, we varied the query frame size q. The results are best for a relatively small query frame, $q = 3, 4, 5$, when the query frame is the same or slightly larger then the context window. The results are worse for a very small query frame, $q = 1, 2$ and for larger query frames $q = 6, 7, ..., 20$ or unlimited. The results presented in the rest of the paper are for $k = 2$ and $q = 5$. For all the other data sets used in this paper (from WSJ 1987 and BNC) we use the parameter values as determined on the development set.

Table 3 shows that the performance is generally better for word counts than for document counts. Therefore, we prefer the method that uses word counts (which is also faster in our particular setting). The difference between them is not statistically significant. Our statistical method performs significantly better than both Edmond's method and the baseline algorithm. For all the results presented in this paper, statistical significance tests were done using the paired t-test, as described in (Manning and Schütze, 1999), page 209.

On average, our method performs 22 percentage points better than the baseline algorithm, and 10 percentage points better than Edmonds's method. Its performance is similar to that of the supervised method (see Section 6.3). An important advantage of our method is that it works on any group of near-synonyms without training, whereas Edmonds's method required a lexical co-occurrence network to be built in advance for each group of near-synonyms and the supervised method required training for each near-synonym group.

We note that the occasional presence of near-synonyms with other senses than the ones we need might make the task somewhat easier. Nonetheless, the task is still difficult, even for human judges, as we will see in Section 6.4. On the other hand, because the solution allows only one correct answer the accuracies are underestimated.

6.2 Comparison to the anti-collocations method

In a second experiment we compare the results of our methods with the anti-collocation method described in Section 4. We use the data set from our previous work, which contain sentences from the first half of the British National Corpus, with near-synonyms from the following eleven groups:

1. benefit, advantage, favor, gain, profit
2. low, gush, pour, run, spout, spurt, squirt, stream
3. deficient, inadequate, poor, unsatisfactory
4. afraid, aghast, alarmed, anxious, apprehensive, fearful, frightened, scared, terror-stricken
5. disapproval, animadversion, aspersion, blame, criticism, reprehension
6. mistake, blooper, blunder, boner, contretemps, error, faux pas, goof, slip, solecism
7. alcoholic, boozer, drunk, drunkard, lush, sot
8. leave, abandon, desert, forsake
9. opponent, adversary, antagonist, competitor, enemy, foe, rival
10. thin, lean, scrawny, skinny, slender, slim, spare, svelte, willowy, wiry
11. lie, falsehood, fib, prevarication, rationalization, untruth

The number of near-synonyms in each group is higher compared with WordNet synonyms, because they are taken from (Hayakawa, 1994), a dictionary that explains differences between near-synonyms. Moreover we retain only the sentences in which at least one of the context words is in our previously acquired knowledge-base of near-synonym collocations. That is, the anti-collocations method works only if we know how a word in the context collocates with the near-synonyms from a group. For the sentences that do not contain collocations or anti-collocations, it will perform no better than the baseline, because the information needed by the method is not available in the knowledge-base. Even if we increase the coverage of the knowledge-base, the anti-collocation method might still fail too often due to words that were not included.

Table 4 presents the results of the comparison. We used two data sets: TestSample, which includes at most two sentences per collocation (the first two sentences from the corpus); and TestAll, which includes all the sentences with collocations as they occurred in the corpus. The reason we chose these two tests is not to bias the results due to frequent collocations.

The last two columns are the accuracies achieved by our method. The second last column shows the results of the method when the word counts are approximated with document counts.

		Accuracy			
Test set	No. of cases	Base-line	Anti-collocs method	Stat. method (Docs)	Stat. method (Words)
Test Sample	171	57.0%	63.3%	**75.6%**	73.3%
TestAll	332	48.5%	58.6%	70.0%	**75.6%**

Table 4: Comparison between the statistical method from Section 3 and the anti-collocations method from Section 4. (Exp2 data set from Section 6.2).

ML method (Weka)	Features	Accuracy
Decision Trees	PMI scores	65.4%
Decision Rules	PMI scores	65.5%
Naïve Bayes	PMI scores	52.5%
K-Nearest Neighbor	PMI scores	64.5%
Kernel Density	PMI scores	60.5%
Boosting (Dec. Stumps)	PMI scores	**67.7%**
Naïve Bayes	500 words	**68.0%**
Decision Trees	500 words	67.0%
Naïve Bayes	PMI + 500 words	66.5%
Boosting (Dec. Stumps)	PMI + 500 words	**69.2%**

Table 5: Comparative results for the supervised learning method using various ML learning algorithms (Weka), averaged over the seven groups of near-synonyms from the Exp1 data set.

The improvement over the baseline is 16 to 27 percentage points. The improvement over the anti-collocations method is 10 to 17 percentage points.

6.3 Comparison to supervised learning

We present the results of the supervised method from Section 5 on the data sets used in Section 6.1. As explained before, the data sets contain sentences with a lexical gap. For each of the seven groups of near-synonyms, the class to choose from, in order to fill in the gaps is one of the near-synonyms in each cluster. We implemented classifiers that use as features either the PMI scores of the left and right context with each class, or the words in the context windows, or both types of features combined. We used as features the 500 most-frequent words for each group of near-synonyms. We report accuracies for 10-fold cross-validation.

Table 5 presents the results, averaged for the seven groups of near-synonyms, of several classifiers from the Weka package (Witten and Frank, 2000). The classifiers that use PMI features are Decision Trees, Decision Rules, Naïve Bayes, K-Nearest Neighbor, Kernel Density, and Boosting a weak classifier (Decision Stumps – which are shallow decision trees). Then, a Naïve Bayes classifier that uses only the word features is presented, and the same type of classifiers with both types of features. The other classifiers from the Weka package were also tried, but the results did not improve and these algorithms had difficulties in scaling up. In particular, when using the 500 word features for each training example, only the Naïve Bayes algorithm was able to run in reasonable time. We noticed that the Naïve Bayes classifier performs very poorly on PMI features only (55% average accuracy), but performs very well on word features (68% average accuracy). In contrast, the Decision Tree classifier performs well on PMI features, especially when using boosting with Decision Stumps. When using both the PMI scores and the word features, the results are slightly higher. It seems that both types of features are sufficient for training a good classifier, but combining them adds value.

		Accuracy		
Test set	Base-line	Supervised Boosting (PMI)	Supervised Boosting (PMI+words)	Unsupervised method
1.	41.7%	55.8%	57.3%	**59.1%**
2.	30.9%	68.1%	**70.8%**	61.5%
3.	70.2%	86.5%	**86.7%**	73.3%
4.	38.0%	66.5%	**66.7%**	66.0%
5.	59.5%	70.4%	71.0%	**72.2%**
6.	36.7%	53.0%	**56.1%**	52.7%
7.	37.0%	74.0%	75.8%	**76.9%**
AVG	44.8%	67.7%	**69.2%**	66.0%

Table 6: Comparison between the unsupervised statistical method from Section 3 and the supervised method described in Section 5, on the Exp1 data set. The results of two of the best supervised classifiers are presented.

Table 6 presents the detailed results of two of the supervised classifiers, and repeats, for easier comparison, the results of the unsupervised statistical method from Section 6.1. The supervised classifier that uses only PMI scores performs similar to the unsupervised method. The best supervised classifier, that uses both types of features, performs slightly better than the unsupervised statistical method, but the difference is not statistically significant. We conclude that the results of the supervised methods and the unsupervised statistical method are similar. An important advantage of the unsupervised method is that it works on any group of near-synonyms without training.

Test set	J1-J2 Agreement	J1 Acc.	J2 Acc.	System Accuracy
1.	72%	70%	76%	53%
2.	82%	84%	84%	68%
3.	86%	92%	92%	78%
4.	76%	82%	76%	66%
5.	76%	82%	74%	64%
6.	78%	68%	70%	52%
7.	80%	80%	90%	77%
AVG	78.5%	79.7%	80.2%	65.4%

Table 7: Results obtained by two human judges on a random subset of the Exp1 data set.

6.4 Results obtained by human judges

We asked two human judges, native speakers of English, to guess the missing word in a random sample of the Exp1 data set (50 sentences for each of the 7 groups of near-synonyms, 350 sentences in total). The judges were instructed to choose words from the list of near-synonyms. The choice of a word not in the list was allowed, but not used by the two judges. The results in Table 7 show that the agreement between the two judges is high (78.5%), but not perfect. This means the task is difficult, even if some wrong senses in the test data might have made the task easier in a few cases.

The human judges were allowed to choose more than one correct answer when they were convinced that more than one near-synonym fits well in the context. They used this option sparingly, only in 5% of the 350 sentences. Taking the accuracy achieved of the human judges as an upper limit, the automatic method has room for improvement (10-15 percentage points). In future work, we plan to allow the system to make more than one choice when appropriate (for example when the second choice has a very close score to the first choice).

7 The intelligent thesaurus

Our experiments show that the accuracy of the first choice being the best choice is 66 to 75%; therefore there will be cases when the writer will not choose the first alternative. But the accuracy for the first two choices is quite high, around 90%, as presented in Table 8.

If the writer is in the process of writing and selects a word to be replaced with a near-synonym proposed by the thesaurus, then only the context on the left of the word can be used for ordering the alternatives. Our method can be easily adapted to consider only the context on the left of the gap. The results of this case are presented in

Test set	Accuracy first choice	Accuracy first 2 choices
Exp1, AVG	66.0%	**88.5%**
Exp2, TestSample	73.3%	**94.1%**
Exp2, TestAll	75.6%	**87.5%**

Table 8: Accuracies for the first two choices as ordered by an interactive intelligent thesaurus.

Test set	Accuracy first choice	Accuracy first 2 choices
Exp1, AVG	58.0%	**84.8%**
Exp2, TestSample	57.4%	**75.1%**
Exp2, TestAll	56.1%	**77.4%**

Table 9: Results of the statistical method when only the left context is considered.

Table 9, for the data sets used in the previous sections. The accuracy values are lower than in the case when both the left and the right context are considered (Table 8). This is due in part to the fact that some sentences in the test sets have very little left context, or no left context at all. On the other hand, many times the writer composes a sentence or paragraph and then she/he goes back to change a word that does not sound right. In this case, both the left and right context will be available.

In the intelligent thesaurus, we could combine the supervised and unsupervised method, by using a supervised classifier when the confidence in the classification is high, and by using the unsupervised method otherwise. Also the unsupervised statistical method would be used for the groups of near-synonyms for which a supervised classifier was not previously trained.

8 Conclusion

We presented a statistical method of choosing the best near-synonym in a context. We compared this method to a previous method (Edmonds's method) and to the anti-collocation method and showed that the performance improved considerably. We also show that the unsupervised statistical method performs comparably to a supervised learning method.

Our method based on PMI scores performs well, despite the well-known limitations of PMI when used with corpora. PMI tends to have problems mostly on very small counts, but it works reasonably with larger counts. Our web corpus is quite large, therefore the problem of small counts does not appear.

In the intelligent thesaurus, we do not make the near-synonym choice automatically, but we let the user choose. The first choice offered by the thesaurus is the best one quite often; the first two choices are correct 90% of the time.

Future work includes a near-synonym sense disambiguation module. In case the target word selected by the writer has multiple senses, they could trigger several groups of near-synonyms. The system will decide which group represents the most likely senses by computing the semantic coherence scores averaged over the near-synonyms from each group.

We plan to explore the question of which inventory of near-synonyms or similar words is the most suitable for use in the intelligent thesaurus.

Choosing the right near-synonym in context is also useful in other applications, such as natural language generation (NLG) and machine translation. In fact we already used the near-synonym choice module in an NLG system, for complementing the choices made by using the symbolic knowledge incorporated into the system.

References

Eneko Agirre and David Martinez. 2000. Exploring automatic word sense disambiguation with decision lists and the Web. In *Proceedings of the Workshop on Semantic Annotation And Intelligent Content, COLING 2000*, Saarbrücken/Luxembourg/Nancy.

Kenneth Church and Patrick Hanks. 1991. Word association norms, mutual information and lexicography. *Computational Linguistics*, 16 (1):22–29.

Charles L. A. Clarke and Egidio Terra. 2003a. Frequency estimates for statistical word similarity measures. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003)*, pages 165–172, Edmonton, Canada.

Charles L. A. Clarke and Egidio Terra. 2003b. Passage retrieval vs. document retrieval for factoid question answering. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 427–428, Toronto, Canada.

Philip Edmonds. 1997. Choosing the word most typical in context using a lexical co-occurrence network. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 507–509, Madrid, Spain.

Gregory Grefenstette. 1999. The World Wide Web as a resource for example-based machine translation tasks. In *Proceedings of the ASLIB Conference on Translating and Computers*, London, UK.

S. I. Hayakawa, editor. 1994. *Choose the Right Word*. Second Edition, revised by Eugene Ehrlich. HarperCollins Publishers.

Diana Inkpen and Graeme Hirst. 2006. Building and using a lexical knowledge-base of near-synonym differences. *Computational Linguistics*, 32 (2):223–262.

Frank Keller and Mirella Lapata. 2003. Using the Web to obtain frequencies for unseen bigrams. *Computational Linguistics*, 29 (3):459–484.

Adam Kilgarriff and Gregory Grefenstette. 2003. Introduction to the special issue on the Web as a corpus. *Computational Linguistics*, 29 (3):333–347.

Adam Kilgarriff. 2001. Web as corpus. In *Proceedings of the 2001 Corpus Linguistics conference*, pages 342–345, Lancaster, UK.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics joint with 17th International Conference on Computational Linguistics (ACL-COLING'98)*, pages 768–774, Montreal, Quebec, Canada.

Christopher Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA.

Rada Mihalcea and Dan Moldovan. 1999. A method for word sense disambiguation from unrestricted text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 152–158, Maryland, MD.

Philip Resnik. 1999. Mining the Web for bilingual text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 527–534, Maryland, MD.

Peter Mark Roget, editor. 1852. *Roget's Thesaurus of English Words and Phrases*. Longman Group Ltd., Harlow, Essex, UK.

P.D. Turney, M.L. Littman, J. Bigham, and V. Shnayder. 2003. Combining independent modules to solve multiple-choice synonym and analogy problems. In *Proceedings of the International Conference RANLP-2003 (Recent Advances in Natural Language Processing)*, pages 482–489, Borovets, Bulgaria.

Peter Turney. 2001. Mining the Web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the Twelfth European Conference on Machine Learning (ECML 2001)*, pages 491–502, Freiburg, Germany.

Ian H. Witten and Eibe Frank. 2000. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco, CA.