Near-Synonym Choice using a 5-gram Language Model

Aminul Islam and Diana Inkpen

University of Ottawa
School of Information Technology and Engineering
Ottawa, ON, Canada, K1N 6N5
{mdislam, diana}@site.uottawa.ca

Abstract. In this work, an unsupervised statistical method for automatic choice of near-synonyms is presented and compared to the state-of-the-art. We use a 5-gram language model built from the Google Web 1T data set. The proposed method works automatically, does not require any human-annotated knowledge resources (e.g., ontologies) and can be applied to different languages. Our evaluation experiments show that this method outperforms two previous methods on the same task. We also show that our proposed unsupervised method is comparable to a supervised method on the same task. This work is applicable to an intelligent thesaurus, machine translation, and natural language generation.

1 Introduction

Choosing the wrong near-synonym can convey unwanted connotations, implications, or attitudes. In machine translation and natural language generation systems, the choice among near-synonyms needs to be made carefully. By *near-synonyms* we mean words that have the same meaning, but differ in lexical nuances. For example, *error*, *mistake*, and *blunder* all mean a generic type of error, but *blunder* carries an implication of *accident* or *ignorance*. In addition to paying attention to lexical nuances, when choosing a word we need to make sure it fits well with the other words in a sentence. In this paper 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 build a strong representation of the context in order to capture the more subtle differences specific to each near-synonym.

The work we present here can be used in an intelligent thesaurus. A writer can access a thesaurus to retrieve words that are similar to a given 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. A standard thesaurus does not offer any explanation about the differences in nuances of meaning between the possible word choices.

This work can also be applied to a natural language generation system [1] that needs to choose among near-synonyms. Inkpen and Hirst [1] included a

Special issue: Natural Language Processing and its Applications.
Research in Computing Science 46, 2010, pp. 41-52

Received 23/11/09
Accepted 16/01/10
Final version 10/03/10

preliminary collocation module that reduces the risk of choosing a near-synonym that does not fit with the other words in a generated sentence (i.e., violates collocational constraints). The work presented in this paper allows for a more comprehensive near-synonym collocation module.

The task we address in this paper is the selection of the best near-synonym that should be used in a particular context. Inkpen [2] argues that 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, Inkpen [2] validates her algorithms by deleting selected words from sample sentences, to see whether the algorithms can restore the missing words. That is, she creates a *lexical gap* and evaluates the ability of the algorithms to fill the lexical gap. Two examples from [2] 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 that can be used to scoring the choices. For our particular task, we choose only the highest scoring near-synonym. 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 of our method, 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 in an automatic way. To verify how difficult the task is for humans, Inkpen [2] performed experiments with human judges on a sample of the test data.

The near-synonym choice method that we propose here uses the Google Web 1T n-gram data set [3], contributed by Google Inc., that contains English word n-grams (from unigrams to 5-grams) and their observed frequency counts calculated over 1 trillion words from web page text collected by Google in January 2006. The text was tokenized following the Penn Treebank tokenization, except that hyphenated words, dates, email addresses and URLs are kept as single tokens. The sentence boundaries are marked with two special tokens <S> and </S>. Words that occurred fewer than 200 times were replaced with the special token <UNK>. Table 1 shows the data sizes of the Web 1T corpus. The n-grams

Table 1. Google Web 1T Data Sizes

Number of	Number	Size on disk (in KB)
Tokens	1,024,908,267,229	N/A
Sentences	95,119,665,584	N/A
Unigrams	13,588,391	185,569
Bigrams	314,843,401	5,213,440
Trigrams	977,069,902	19,978,540
4-grams	1,313,818,354	32,040,884
5-grams	1,176,470,663	33,678,504

themselves must appear at least 40 times to be included in the Web 1T corpus[1]. It is expected that this data will be useful for statistical language modeling, e.g., for machine translation or speech recognition, as well as for other uses.

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

---

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

This paper is organized as follow: Section 2 presents a brief overview of the related work. Our proposed method is described in Section 3. Evaluation and experimental results are discussed in Section 4. We conclude in Section 5.

2 Related Work

The idea of using the Google Web 1T n-gram data set as a resource in different natural language processing applications has been exploited by many researchers. Islam and Inkpen [4] use 3-grams of this data set to detect and correct real-word spelling errors and also use n-grams to only correct real-word spelling errors [5]. Nulty and Costello [6] deduce the semantic relation that holds between two nouns in a noun-noun compound phrase such as "flu virus" or "morning exercise" using lexical patterns in the Google Web 1T corpus. Klein and Nelson [7] investigate the relationship between *term count* (*TC*) and *document frequency* (*DF*) values of terms occurring in the Web as Corpus (WaC) and also the similarity between *TC* values obtained from the WaC and the Google n-gram dataset and they mention that a strong correlation between the two would give them confidence in using the Google n-grams to estimate accurate inverse document frequency (IDF) values in order to generate well-performing lexical signatures based on the TF-IDF scheme. Murphy and Curran [8] explore the strengths and limitations of Mutual Exclusion Bootstrapping (MEB) by applying it to two novel lexical-semantic extraction tasks: extracting bigram named entities and WordNet lexical file classes [9] from the Google Web 1T 5-grams.

Turney *et al.* [10] addressed the multiple-choice synonym problem: given a word, choose a synonym for that word, among a set of possible solutions. In

[1] Details of the Google Web 1T data set can be found at www.ldc.upenn.edu/Catalog/docs/LDC2006T13/readme.txt

this case the solutions contain one synonym and some other (unrelated) words. They achieves high performance by combining classifiers. Clarke and Terra [11] 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.

In fact, the works that address exactly the same task are that of Edmonds [12] and Inkpen [2], as far as we are aware. Edmonds [12] gives a solution based on a lexical co-occurrence network that included second-order co-occurrences whereas Inkpen [2] uses a much larger corpus and a simpler method, and obtains better results than that of [12].

Inkpen's [2] unsupervised method is based on the mutual information scores between a near-synonym and the content words in the context filtering out the stopwords[2]. 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) [13].

$$PMI(x, y) = \log_2 \frac{P(x, y)}{P(x)P(y)}$$

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 denote frequency counts and N is the total number of words in the corpus. Therefore,

$$PMI(x, y) = \log_2 \frac{C(x, y) \cdot N}{C(x) \cdot C(y)}$$

where N can be ignored in comparisons, since is it the same in all the cases. Inkpen [2] models 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 $s = \cdots w_1 \cdots w_k \quad Gap \quad w_{k+1} \cdots w_{2k} \cdots$, for each near-synonym NS_i from the group of candidates, the score is computed by the following formula:

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

In a supervised learning method, Inkpen [2] trains classifiers for each group of near-synonyms. The classes are the near-synonyms in the solution set. Each sentence is converted into a vector of features to be used for training the supervised classifiers. Inkpen used two types of features. One type of features consists in the scores of the left and right context with each class (i.e., with each near-synonym from the group). The number of features of this type is equal to twice the number of classes: one feature for the score between the near-synonym and

[2] We do not filter out stopwords or punctuation in our method.

the part of the sentence at the left of the gap, and one feature for the score between the near-synonym and the part of the sentence at the right of the gap. The second type of features is formed by the words in the context windows. For each group of near-synonyms, Inkpen 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. Inkpen trained classifiers using several machine learning algorithms to see which one is best at discriminating among the near-synonyms.

There has been quite a lot of work in unsupervised learning of word clusters based on n-grams [14–16]. 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.

3 Proposed Method

Our task is to find the best near-synonym from a set of candidates that could fill in the gap in an input text, using the Google Web 1T data set. Let us consider an input text W which after tokenization[3] has p ($2 \leq p \leq 9$) words[4], i.e.,

$$W = \{\ldots\ w_{i-4}\ w_{i-3}\ w_{i-2}\ w_{i-1}\ \boxed{w_i}\ w_{i+1}\ w_{i+2}\ w_{i+3}\ w_{i+4}\ \ldots\}$$

where $\boxed{w_i}$ (in position i) indicates the gap and w_i in $\boxed{w_i}$ denotes a set of m near-synonyms (i.e., $w_i = \{s_1, s_2, \cdots, s_j, \cdots, s_m\}$). We take into account at most four words before the gap and at most four words after the gap. Our task is to choose the $s_j \in w_i$ that best matches with the context. In other words, the position i is the gap that needs to be filled with the best suited member from the set, w_i.

[3] We need to tokenize the input sentence to make the n-grams formed using the tokens returned after the tokenization consistent with the Google n-grams. The input sentence is tokenized in a manner similar to the tokenization of the Wall Street Journal portion of the Penn Treebank. Notable exceptions include the following:

- Hyphenated word are usually separated, and hyphenated numbers usually form one token.
- Sequences of numbers separated by slashes (e.g., in dates) form one token.
- Sequences that look like urls or email addresses form one token.

[4] If the input text has more than 9 words then we keep at most four words before the gap and at most four words after the gap to make the length of the text 9. We choose these numbers so that we could maximize the number of n-grams to use, given that we have up to 5-grams in the Google Web 1T data set.

We construct m strings $(S_1 \cdots S_m)$ replacing the gaps in position i with $s_j \in w_i$ as follows:

$$S_1 = \cdots w_{i-4} w_{i-3} w_{i-2} w_{i-1} s_1 w_{i+1} w_{i+2} w_{i+3} w_{i+4} \cdots$$

$$S_2 = \cdots w_{i-4} w_{i-3} w_{i-2} w_{i-1} s_2 w_{i+1} w_{i+2} w_{i+3} w_{i+4} \cdots$$

$$\vdots$$

$$S_m = \cdots w_{i-4} w_{i-3} w_{i-2} w_{i-1} s_m w_{i+1} w_{i+2} w_{i+3} w_{i+4} \cdots$$

Using equation 7 in Section 3.2, we calculate $P(S_1), \cdots, P(S_m)$. The index of the target synonym, j, will be $\underset{j \in 1 \cdots m}{\operatorname{argmax}} P(S_j)$.

3.1 n-gram Language Model

A *language model* is usually formulated as a probability distribution $P(S)$ over strings S, and attempts to reflect how frequently a string S occurs as a sentence. The most widely-used language models, by far, are n-gram language models [17]. We introduce these models by considering the case $n = 5$; these models are called 5-gram models. For a sentence S composed of the words $w_1 \cdots w_p$, without loss of generality we can express $P(S)$ as

$$\begin{aligned} P(S) &= P(w_1)P(w_2|w_1)P(w_3|w_1w_2)\cdots P(w_p|w_1\cdots w_{p-1}) \\ &= \prod_{i=1}^{p} P(w_i|w_1\cdots w_{i-1}) \end{aligned} \tag{1}$$

For n-gram models where $n > 2$, we condition the probability of a word on the identity of the last n-1 words. Generalizing equation 1 to $n > 2$, we get

$$P(S) = \prod_{i=1}^{p+1} P(w_i|w_{i-n+1}^{i-1}) \tag{2}$$

where w_i^j denotes the words $w_i \cdots w_j$. To estimate $P(w_i|w_{i-n+1}^{i-1})$, the frequency with which the word w_i occurs given that the words w_{i-n+1}^{i-1} precede the current word w_i, we simply count how often the n-gram w_{i-n+1}^{i} occurs in some text and normalize by the total number of occurrences of any word in the same context. Let $C(w_{i-n+1}^{i})$ denote the number of times the n-gram w_{i-n+1}^{i} occurs in the given text. Then

$$P(w_i|w_{i-n+1}^{i-1}) = \frac{C(w_{i-n+1}^{i})}{\sum_{w_i} C(w_{i-n+1}^{i})} \tag{3}$$

Notice that $C(w_{i-n+1}^{i-1}) \geq \sum_{w_i} C(w_{i-n+1}^{i})$. To understand the inequality of $C(w_{i-n+1}^{i-1})$ and $\sum_{w_i} C(w_{i-n+1}^{i})$, assume that both i and n are 5. Then, $C(w_{i-n+1}^{i-1})$ becomes $C(w_1w_2w_3w_4)$, which is actually the frequency of a specific 4-gram, $w_1w_2w_3w_4$, and $\sum_{w_i} C(w_{i-n+1}^{i})$ becomes $\sum_{w_5} C(w_1w_2w_3w_4w_5)$, which is the

sum of the frequencies of all the 5-grams that start with the 4-gram, $w_1w_2w_3w_4$. Thus, in general, $C(w_1w_2w_3w_4)$ is equal to $\sum_{w_5} C(w_1w_2w_3w_4w_5)$. But, for some specific cases, it is possible that $C(w_1w_2w_3w_4)$ is greater than $\sum_{w_5} C(w_1w_2w_3w_4w_5)$. For example, all the n-grams ($2 \leq n \leq 5$) that appear less than 40 times have been filtered out from the Google Web 1T n-grams. This means all the 5-grams (starting with the 4-gram $w_1w_2w_3w_4$) that appear less than 40 times have not been included in $\sum_{w_5} C(w_1w_2w_3w_4w_5)$. Thus, when we deal with the Web 1T 5-grams and 4-grams, it is obvious that $C(w_1w_2w_3w_4)$ is greater than or equal to $\sum_{w_5} C(w_1w_2w_3w_4w_5)$. Thus, in general, we can say that $C(w_{i-n+1}^{i-1}) \geq \sum_{w_i} C(w_{i-n+1}^{i})$. This also supports the idea of using the missing count (equation 4) in the smoothing formula (equation 5).

We use a smoothing method loosely based on the *one-count* method described in [18]. Because tokens that appears less than 200 times and n-grams that appear less than 40 times have been filtered out from the Web 1T, we use n-grams with missing counts instead of n-grams with one counts [19]. The missing count is defined as:

$$M(w_{i-n+1}^{i-1}) = C(w_{i-n+1}^{i-1}) - \sum_{w_i} C(w_{i-n+1}^{i}) \tag{4}$$

The corresponding smoothing formula is:

$$P(w_i|w_{i-n+1}^{i-1}) = \frac{C(w_{i-n+1}^{i}) + (1+\alpha_n)M(w_{i-n+1}^{i-1})P(w_i|w_{i-n+2}^{i-1})}{C(w_{i-n+1}^{i-1}) + \alpha_n M(w_{i-n+1}^{i-1})} \tag{5}$$

Yuret [19] optimized the parameters $\alpha_n > 0$ for $n = 2 \cdots 5$ on the Brown corpus to yield a cross entropy of 8.06 bits per token. The optimized parameters are: $\alpha_2 = 6.71$, $\alpha_3 = 5.94$, $\alpha_4 = 6.55$, $\alpha_5 = 5.71$

Thus, incorporating the smoothing formula in equation 2, we get

$$P(S) = \prod_{i=1}^{p+1} \frac{C(w_{i-n+1}^{i}) + (1+\alpha_n)M(w_{i-n+1}^{i-1})P(w_i|w_{i-n+2}^{i-1})}{C(w_{i-n+1}^{i-1}) + \alpha_n M(w_{i-n+1}^{i-1})} \tag{6}$$

3.2 The Language Model Used for Our Task

For the specified task, we simplify the 5-gram model described in Section 3.1. From equation 2, it is clear that the maximum number of products possible is 10 as $2 \leq p \leq 9$. Among these products, we can omit the products $P(w_{i-1}|w_{i-5}^{i-2})$, $P(w_{i-2}|w_{i-6}^{i-3})$, $P(w_{i-3}|w_{i-7}^{i-4})$, $P(w_{i-4}|w_{i-8}^{i-5})$, and $P(w_{i+5}|w_{i+1}^{i+4})$ because these product items have the same values for all $j \in 1 \cdots m$. Thus, the five product items that we consider are: $P(w_i|w_{i-4}^{i-1})$, $P(w_{i+1}|w_{i-3}^{i})$, $P(w_{i+2}|w_{i-2}^{i+1})$, $P(w_{i+3}|w_{i-1}^{i+2})$, and $P(w_{i+4}|w_{i}^{i+3})$. Applying this simplification in equation 2 and equation 6, we get

$$\begin{aligned} P(S) &= \prod_{i=5}^{p} P(w_i|w_{i-n+1}^{i-1}) \\ &= \prod_{i=5}^{p} \frac{C(w_{i-n+1}^{i}) + (1+\alpha_n)M(w_{i-n+1}^{i-1})P(w_i|w_{i-n+2}^{i-1})}{C(w_{i-n+1}^{i-1}) + \alpha_n M(w_{i-n+1}^{i-1})} \end{aligned} \tag{7}$$

Equation 7 is actually used in a recursive way for n=5,4,3,2,1 (i.e., if the current n-gram count is zero, then it is used with a lower n-gram, and so on). For example, $P(w_5|w_1w_2w_3w_4)$ is a function of $C(w_1w_2w_3w_4w_5)$ and $P(w_5|w_2w_3w_4)$; if $C(w_1w_2w_3w_4w_5) > 0$ then we do not consider $P(w_5|w_2w_3w_4)$. This is a backoff language model.

4 Evaluation and Experimental Results

4.1 Comparison to Edmonds's and Inkpen's methods

In this section we present results of the proposed method explained in Section 3. We compare our results with those of Inkpen [2] and Edmonds [12]. Edmonds [12] 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. Inkpen's [2] method is based on mutual information, not on co-occurrence counts. Inkpen's counts are collected from a much larger corpus.

Test Set	Number of Cases	Accuracy				
		Base Line	Edmonds's Method	Inkpen's Method (Supervised)	Inkpen's Method (Unsup.)	Proposed Method (Unsup.)
difficult, hard, tough	6,630	41.7%	47.9%	57.3%	59.1%	**63.2%**
error, mistake, oversight	1,052	30.9%	48.9%	70.8%	61.5%	**78.7%**
job, task, duty	5,506	70.2%	68.9%	**86.7%**	73.3%	78.2%
responsibility, burden, obligation, commitment	3,115	38.0%	45.3%	66.7%	66.0%	**72.2%**
material, stuff, substance	1,715	59.5%	64.6%	71.0%	**72.2%**	70.4%
give, provide, offer	11,504	36.7%	48.6%	**56.1%**	52.7%	55.8%
settle, resolve	1,594	37.0%	65.9%	75.8%	**76.9%**	70.8%
ALL (average over all sentences)	31,116	44.9%	53.5%	65.2%	61.7%	**65.3%**
ALL (average from group averages)	31,116	44.8%	55.7%	69.2%	66.0%	**69.9%**

Table 2. Comparison among the new proposed method, a baseline algorithm, Edmonds's method, and Inkpen's unsupervised and supervised method

For comparison purposes, in this section we use the same test data (WSJ 1987) and the same groups of near-synonyms. The seven groups of near-synonyms used by Edmonds are listed in the first column of Table 2. 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.

Before we look at the results, we mention that the accuracy values we compute are the percentage of correct choices when filling in the gap with the winning near-synonym. The expected solution is the near-synonym that was originally in the sentence, and it was taken out to create the gap. This measure is conservative; it does not consider cases when more than one solution is correct.

Table 2 presents the comparative results for the seven groups of near-synonyms. The second last row averages the accuracies for all the test sentences, i.e., these are calculated as the number of correct choices returned over total number of sentences (i.e., 31116). The last row averages the accuracies for all the groups averages, i.e., these are calculated as the sum of the accuracies (in percentage) of all the seven groups over the number of groups (i.e., 7). 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 [12]. The fifth column presents the result of Inkpen's [2] supervised method when using boosting (decision stumps) as machine learning method and PMI+500 words as features. The sixth column presents the result of Inkpen's [2] unsupervised method when using word counts in PMI, and the last column is for our proposed unsupervised method using the Google n-grams. We show in bold the best accuracy figure for each data set. We notice that the automatic choice is more difficult for some near-synonym groups than for the others.

Our method performs significantly better than the baseline algorithm, Edmond's method, and Inkpen's unsupervised method and comparable to Inkpen's supervised method. For all the results presented in this paper, statistical significance tests were done using the paired t-test, as described in [20], page 209. Error analysis reveals that incorrect choices happen more often when the context is weak, that is, very short sentences or sentences with very few content words.

On average, our method performs 25 percentage points better than the baseline algorithm, 14 percentage points better than Edmonds's method, 4 percentage points better than Inkpen's unsupervised method, and comparable to Inkpen's supervised method. An important advantage of our method is that it works on any group of near-synonyms without training, whereas Edmonds's method requires a lexical co-occurrence network to be built in advance for each group of near-synonyms and Inkpen's supervised method requires training for each near-synonym group.

Some examples of correct and incorrect choices, using our proposed method are shown in Table 3. Table 4 shows some examples of cases where our proposed method fails to generate any suggestion. Cases where our method failed to provide any suggestion are due to the appearances of some very uncommon proper names or nouns, contractions (e.g., n't), hyphens between two words (e.g., teen-agers), single and double inverted commas in the n-grams, and so on. Preprocessing to tackle this issues would improve the results.

CORRECT CHOICE:
... viewed widely as a *mistake* → mistake [error, mistake, oversight] and a major ...
... analysts expect stocks to *settle* → settle [settle, resolve] into a steady ...
... Sometimes that *task* → task [job, task, duty] is very straightforward ...
... carry a heavier tax *burden* → burden [responsibility, burden, obligation, commitment] during 1987 because ...
INCORRECT CHOICE:
... would be a political *mistake* → error [error, mistake, oversight] to criticize the ...
... its energies on the *material* → substance [material, stuff, substance] as well as ...
... 23 , and must *provide* → give [give, provide, offer] Washington-based USAir at ...
... Phog Allen – to *resolve* → settle [settle, resolve] the burning question ...

Table 3. Examples of correct and incorrect choices using our proposed method. Italics indicate the proposed near-synonym choice returned by the method, arrow indicates the original near-synonym, square brackets indicate the solution set.

NO CHOICE:
... two exchanges ' " → commitment [responsibility, burden, obligation, commitment] to making serious ...
... He said ECD 's → material [material, stuff, substance] is a multicomponent ...
... Safe Rides , teen-agers → give [give, provide, offer] rides to other ...
... guilty plea does n't → resolve [settle, resolve] the issue for ...
... sees Mr. Haig 's → tough [difficult, hard, tough] line toward Cuba ...
... The 1980 Intelligence → Oversight [error, mistake, oversight] Act requires that ...
... thought Mr. Cassoni 's → job [job, task, duty] will be made ...

Table 4. Examples of sentences where our method fails to generate any suggestion.

4.2 Experiment with human judges

Inkpen [2] asked two human judges, native speakers of English, to guess the missing word in a random sample of the experimental data set (50 sentences for each of the 7 groups of near-synonyms, 350 sentences in total). The results in Table 5 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 automatically-produced test data might have made the task easier in a few cases.

[5] Here, each of the seven groups has equal number of sentences, which is 50. Thus, the average from all 350 sentences and the average from group averages are the same.

Test set	J1-J2 Agreement	J1 Accuracy	J2 Accuracy	Inkpen's Accuracy	Our Accuracy
difficult, hard, tough	72%	70%	76%	53%	62%
error, mistake, oversight	82%	84%	84%	68%	70%
job, task, duty	86%	92%	92%	78%	80%
responsibility, burden, obligation, commitment	76%	82%	76%	66%	76%
material, stuff, substance	76%	82%	74%	64%	56%
give, provide, offer	78%	68%	70%	52%	52%
settle, resolve	80%	80%	90%	77%	66%
All (average)[5]	78.5%	79.7%	80.2%	65.4%	66%

Table 5. Experiments with two human judges on a random subset of the experimental data set

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. In future work, we plan to allow the system to make more than one choice when appropriate (e.g., when the second choice has a very close score to the first choice).

5 Conclusions

We presented an unsupervised statistical method of choosing the best near-synonym in a context. We compared this method with three previous methods (Edmonds's method and two of Inkpen's method) and show that the performance improved considerably. It is interesting that our unsupervised statistical method is comparable to a supervised learning method.

Future work includes an intelligent thesaurus and a natural language generation system that has knowledge of nuances of meaning of near-synonyms. We plan to include a near-synonym sense disambiguation module to ensure that the thesaurus does not offer alternatives for wrong senses of words.

References

1. Inkpen, D.Z., Hirst, G.: Near-synonym choice in natural language generation. In: Proceedings of the International Conference RANLP-2003 (Recent Advances in Natural Language Processing), Borovets, Bulgaria (2003) 204–211
2. Inkpen, D.: A statistical model for near-synonym choice. ACM Transactions on Speech and Language Processing 4 (2007) 1–17
3. Brants, T., Franz, A.: Web 1T 5-gram corpus version 1.1. Technical report, Google Research (2006)

4. Islam, A., Inkpen, D.: Real-word spelling correction using Google Web 1T 3-grams. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, Singapore, Association for Computational Linguistics (2009) 1241–1249
5. Islam, A., Inkpen, D.: Real-word spelling correction using Google Web 1T n-gram data set. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, ACM (2009) 1689–1692
6. Nulty, P., Costello, F.: Using lexical patterns in the Google Web 1T corpus to deduce semantic relations between nouns. In: Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009), Boulder, Colorado, Association for Computational Linguistics (2009) 58–63
7. Klein, M., Nelson, M.L.: Correlation of term count and document frequency for Google n-grams. In et al., B.M., ed.: Proceedings of the 31st European Conference on Information Retrieval. Volume 5478/2009 of Lecture Notes in Computer Science., Springer Berlin / Heidelberg (2009) 620–627
8. Murphy, T., Curran, J.: Experiments in mutual exclusion bootstrapping. In: Proceedings of the Australasian Language Technology Workshop 2007, Melbourne, Australia (2007) 66–74
9. Fellbaum, C., ed.: WordNet: An electronic lexical database. MIT Press (1998)
10. Turney, P., Littman, M., Bigham, J., Shnayder, V.: 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), Borovets, Bulgaria (2003) 482–489
11. Clarke, C.L.A., Terra, E.: 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), Edmonton, Canada (2003) 165–172
12. Edmonds, P.: 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, Madrid, Spain (1997) 507–509
13. Church, K., Hanks, P.: Word association norms, mutual information and lexicography. Computational Linguistics **16 (1)** (1991) 22–29
14. Brown, P.F., deSouza, P.V., Mercer, R.L., Pietra, V.J.D., Lai, J.C.: Class-based *n*-gram models of natural language. Computational Linguistics **18** (1992) 467–479
15. Pereira, F., Tishby, N., Lee, L.: Distributional clustering of english words. In: Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics. (1993) 183–190
16. Lin, D.: Automatic retrieval and clustering of similar words. In: Proceedings of the 17th international conference on Computational linguistics, Morristown, NJ, USA, Association for Computational Linguistics (1998) 768–774
17. Chen, S.F., Goodman, J.T.: An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Computer Science Group, Harvard University (1998)
18. Chen, S.F., Goodman, J.: An empirical study of smoothing techniques for language modeling. In: 34th Annual Meeting of the Association for Computational Linguistics. (1996) 310–318
19. Yuret, D.: KU: Word sense disambiguation by substitution. In: Proceedings of the 4th workshop on Semantic Evaluations (SemEval-2007), Prague, Czech Republic (2007) 207–214
20. Manning, C., Schütze, H.: Foundations of Statistical Natural Language Processing. The MIT Press, Cambridge, Massachusetts (1999)