Comparison of Some Global Coherence Optimization Heuristics for Word Sense Disambiguation

ALEXANDER GELBUKH,[1] GRIGORI SIDOROV,[1] and SAN-YONG HAN[2]
[1] Natural Language and Text Processing Laboratory,
Center for Computing Research, National Polytechnic Institute,
Av. Juan Dios Batiz s/n, Zacatenco 07738, DF
MEXICO
{gelbukh, sidorov}@cic.ipn.mx, www.gelbukh.com
[2] Chung-Ang University, Seoul,
KOREA
hansy@cau.ac.kr

EXTENDED ABSTRACT

Note: The Call for Papers specified that for evaluation extended abstracts or paper drafts should be submitted.

ABSTRACT. For most English words dictionaries give various senses: e.g., "*bank*" can stand for a financial institution, shore, set, etc. Automatic selection of the sense intended in a given text has crucial importance in many applications of text processing, such as information retrieval or machine translation: e.g., "(*my account in the*) *bank*" is to be translated into Spanish as "(*mi cuenta en el*) *banco*" whereas "(*on the*) *bank* (*of the lake*)" as "(*en la*) *orilla* (*del lago*)." To choose the optimal combination of the intended senses of all words, the global coherence of the text is to be considered. By coherence, we mean average relatedness between the chosen senses for all words in the text. Due to high dimensionality of the search space, heuristics are to be used to find a near-optimal configuration. In this paper, we compare several such heuristics and discuss their advantages and disadvantages in terms of complexity and quality of the results.

1 Introduction

Nearly any word we use in everyday communication has several possible interpretations, called senses. For example, the word *bank* can be interpreted as a financial institution, river shore, stock of some objects, etc. The word *bill* can be interpreted as a banknote, law proposal, mouth of a bird, agricultural tool, etc. The very word *word* can be interpreted as a unit of speech or text, advice, message, promise, password, a unit of computer memory, a unit of DNA sequence, etc. If you take any word that comes to your mind and look it up in a dictionary, you will probably be surprised with the number of different interpretations it allows.

Obviously, for correct understanding of a text, the reader—be it a human being or a computer program—must be able to determine what sense is intended for each word in the text. Indeed, if I advise you to keep your money in a reliable bank, would you carry it to a financial institution or a river shore? If I tell this to a robot, wouldn't it choose a river shore?

Apart from message understanding, there are a number of important applications where automatically determining the correct sense of a word is crucial. One of them is information retrieval. Suppose the user of an Internet search engine such as Google types in query asking for banks in Chicago. First, the program should clarify which sense of *bank* the user needs—for example, asking the user to choose it from a menu; suppose the user chooses a financial institute. Now for each document containing the word *bank* the program has to automatically decide whether it is the requested sense that appears in the text. For example, of the following two texts:

- *There are two Citybank branch banks in the central area of Chicago,*
- *The hotel is located at the beautiful bank Michigan Lake, with an amazing view Chicago's skyscrapers*

cuando lo usé para levantar mi carro: indeed, the Spanish word *gato* has senses *cat* and *jack* (used to jack up cars). If you have ever used an automatic translation system, you perhaps have noticed hundreds such errors.

A typical explanatory dictionary lists for each word its different senses. Words having more than one sense are called polysemous; in fact, nearly any word appearing in a text is polysemous. Given a specific dictionary and a specific occurrence of a word in a specific text, the problem of the choice, out of the senses listed for this word in this dictionary, of the one intended for this occurrence is called the word sense disambiguation (WSD) problem [4]. A typical WSD program takes in input a text and copies it to its output, marking each polysemous word with the intended sense according to the explanatory or bilingual dictionary used. E.g., with the dictionary [1] the output for the first text in our example above would be *There*$_2$ *are*$_1$ *two*$_1$ *Citybank branch*$_3$ *banks*$_{2\text{-}1}$ *in*$_{1\text{-}2}$ *the*$_{1\text{-}1}$ *central*$_1$ *area*$_2$ *of*$_{2\text{-}1}$ *Chicago*.

In spite of the great attention the problem has received in the last years and important developments achieved, the precision of the state-of-the art algorithms is far from being satisfactory. In this paper, we consider a new WSD technique that we have introduced in our previous work [6], consisting in global optimization of text coherence using a genetic algorithm. By text coherence we mean the total word sense relatedness in the text: the more related the words in the text to each other the better the coherence of the text.

The paper is organized as follows. Section 2 presents the related work. Section 3 introduces the word relatedness measure, which is the function under optimization in this paper. Section 4 explains the basic terms. Section 5 formulates the problem. Section 6 explains the heuristics we compare for the solution of this problem. Section 7 presents the experimental results and discussion. Finally, Section 8 concludes the paper.

2 Related Work

Many methods in WSD and similar tasks are based on optimization of some word relatedness measure, which gives a numerical estimate of the probability of two words (or word senses) to appear in the same text fragment [7], [11]; the senses are chosen that are more probable in a given context. Padwardhan *et al.* [12] have compared different such measures and reported the Lesk relatedness measure [3], [11] to be one of the most promising one, so that it is this measure that we have chosen for our experiments. It is based on the use of existing explanatory dictionaries, see more details in Section 3. In fact, however, our method does not rely on a specific word relatedness measure, and in the future we plan to experiment with other measures, too.

Word relatedness can be measured between two senses of a word or between a word sense and a lexeme. For example, one can measure the relatedness between the sense *bank*$_2$ 'financial institute' and the sense *branch*$_3$ 'office' or between the sense *bank*$_2$ 'financial institute' and the lexeme *branch*, without specifying its specific sense ('stick', 'science', 'office', 'offspring', 'ramification', 'power', 'instruction', etc.).

Given a word relatedness measure, the problem is to choose the senses for each word in such a manner that increases the word relatedness. This can be done in a local or global manner. The existent algorithms use the local optimization: for each word, the sense that has better relatedness with the surrounding words is chosen. The relatedness is measured between each sense of the given word and the context words considered as lexemes but not specific senses. No attempt is made to choose a combination of senses in the whole text that globally optimizes the relatedness between all senses.

Araujo [2] described a method of global optimization for a similar disambiguation problem, namely, part-of-speech (POS) tagging. She used a genetic algorithm [10] to find the best combination of POS tags in a sentence. Our method is inspired by that work.

3 Word Relatedness Measure

There are many possible word relatedness measures [12]. Here we introduce the Lesk relatedness measure we used for our experiments.

We consider words senses as definitions in an explanatory dictionary. With this, the senses are treated as short texts (namely, definitions) in the same language as the word under consideration. Moreover, we use a "bag-of-words" approach, i.e., we reduce such definitions to sets of lexemes. The latter term refers to morphological normalization (stemming): different morphological forms of the textual words reduced to a common root; for example, *give*, *gives*, *gave*, *given* are the forms of the same lexeme *to give*. Such a reduction is performed during data preprocessing [5].

For estimating relatedness between two sets of words, we use a measure analogous to the Dice coefficient [8], [9] that gives a numerical estimation of the degree of intersection between two sets. The simplest way to measure such intersection is to calculate the number of common words in the two sets; this is referred to as the Lesk measure. As was mentioned, the occurrences of different wordforms of the same lexeme (root, or stem) in the two texts are counted as intersections, as if they were the same word. Recently we have improved Lesk measure to also take into account the synonyms of words in the two sets [13]. Note that we do not consider the synonyms as an additional "weak" knowledge source as in [14], but the synonyms are treated as textual intersections. In our opinion, the basis of this idea is that the synonyms express practically the same concept, and the differences in shades of meaning can be ignored in our tasks.

In Lesk algorithm [11], the intersection is measured between the definition of a sense of the word (considered as a bag of words) and the context of the specific occurrence of the given word, again considered as a bag of words. Note that in this case what is compared is a specific sense of a word and a lexeme without specifying its sense, see discussion in Section 2. To form the word set representing a lexeme, the dictionary definitions of all its senses are joined together. Similarly, in case of morphological or POS ambiguity of a word in the context, in our experiments we joined together the definitions for all its possible interpretations.

In our method, however, we only use the relatedness between two specific senses of words and not between a sense of a given word and the lexemes (joint senses) of its surrounding context.

One should distinguish between relatedness of two words or word senses in language and in a specific text. Relatedness in language expresses the possibility of two words to be used in a description of the same situation. Meanwhile, relatedness in a text expresses the plausibility of the hypotheses that these two particular word occurrences are actually used in a description of the same situation. While the former type of relatedness does not depend on the position of the words in any particular text, the latter one generally decreases with the linear distance between the words: the words used far from each other are hardly related to the description of the same situation.

Accordingly, we smooth the relatedness measure depending on the distance, considering it to be zero if the distance between the two words exceeds a certain threshold (text window size).

4 The Data Structure

The numerical problem at hand, we use the following data structure:

- The whole text is subdivided into a set of text fragments, which can correspond to sentences, paragraphs, sections, etc. Currently we consider the whole text as one long fragment.
- A text fragment *f* is a sequence of words.
- A word *w* is a set of word senses.
- The relatedness between individual senses of individual word occurrences is given by a matrix M (s_i, s_j), where s_k is a specific sense of a specific word occurrence in the text. The relatedness is not defined between senses of the same word occurrence.

The relatedness can depend on the linear syntactic distance between words, which is the reason to consider word occurrences and not just word types.

5 The Problem

The problem consists in selecting, for each word, one sense that is more likely to be the intended sense the given text. Various heuristics can be applied to find a plausible combination of choices. In other words, upon the described numerical representation this weakly formulated problem can be formalized in various ways.

The standard Lesk algorithm is formalized follows: for each word, select the sense that has maximum average relatedness to the nearby words. Here is the algorithm solving this problem:

```
for each word $w$
    for each sense $s$
        $score(s) = \sum_{i,j} M_{ij}(w,s)$        (1)
    select $s_{best}$ = max arg ($score(s)$)
```

We do not normalize the score to calculate the average since this does not affect the result. In case of equal scores for two or more senses, we choose the first one of them.

This approach is based on the hypothesis that the correct sense of the word w_i is not known and the probability for the sense to be the intended one distributed uniformly. The average relatedness is the relatedness of a given sense to the senses of the other word weighted by the probability of those senses.

However, this idea suffers from a logical inconsistency: upon termination of the algorithm, only one sense of each nearby word is selected. Thus, the choice of the sense for the word *w* is affected by the words w_i's senses that will be rejected (or even have been rejected) by the same algorithm at other steps.

A logically consistent problem formulation is follows: choose such the combination of choices senses of all words in the text with maximum average distance between the senses. If there are several such combinations, choose one of them.

Such a task is consistent because we take into account only the distances between the senses that, according our choice, are actually present in the given text.

This task is logically equivalent to the following task (since the task is solved independently for each fragment, we consider only one fragment). Let *N* be number of words in the text fragment and n_i be number of senses of the word w_i. Denote **N** the set natural numbers. A combination of choices of senses a function

$$f : \{1, ..., N\} \rightarrow \mathbf{N}$$

such that $1 \le f(i) \le n_i$. Denote **F** the set of all such functions. An imaginable algorithm for solving the task as follows:

```
for each sequence $f \in \mathbf{F}$
    for each word $w_k$
        $score(w_k) = \sum_l M_{l,f(w_l)}(w, f(k))$
    $score(f) = \sum_k^N score(w_k)$        (3)
$f_{best}$ = max arg ($score(f)$)
for each word $w_k$
    select $s_{best} = f_{best}(k)$
```

The algorithm consists of finding such a way *f* of assigning senses to words that maximizes the average relatedness *score*(w_k) between these senses. Note that the senses other than the selected ones are not involved in the process.

The size of the search space is

$$|\mathbf{F}| = \prod_l^N n_l \, .$$

Consider a text of $N = 1000$ words, such that each second word has at least 3 senses. Then the search space is $3^{500} = 3 \times 10^{238}$. This is not an exaggerated example: in a randomly selected text used for our experiments, the average number of senses per word was 3.82 (750 senses in total by 196 words); with this text the search space was about 1.7×10^{114}.

Thus, heuristic methods are necessary to find the best combination.

6 The Heuristics

To select the best sequence, a number of heuristics can be tried. In fact, the problem strongly resembles such problems as the traveling salesman problem and the like, so the heuristics we suggest resemble those used for such kind of problems.

Genetic algorithm. We have reported an application of the genetic algorithm to this problem in our previous work [6].

The cautious algorithm described above.

A greedy algorithm using the formula

$$score(s) = \sum_l \max_j M_{lj}(w, s) \, . \qquad (2$$

Ordering selection algorithm (corresponding to the insertion algorithm for the traveling salesman problem): choose the order in which the decision on individual word occurrences are made, and after each decision fix the sense of the given word,

as well as a number of other heuristics. In the case of ordering selection algorithm, different heuristics can be used to choose the optimal order:

- Direct order: from the first to the last word,
- Inverse order: from the last to the first word,
- Greedy insertion order: at each step, choose the next element that gives the best increase in the total coherence value,
- A genetic algorithm can be used to find an optimal order.

7 Experimental Results and Discussion

We experimented with a 196 words long Spanish text taking from an Internet news site. We built the sense relatedness matrix according to the Lesk methodology: the relatedness is the Dice coefficient calculated for the intersection of the definitions of the two terms.

We have implemented the algorithms enumerated above and are finishing their comparison (our experiments are still in progress). The actual experimental results will be given in detail in the final version of this paper, but we do not give them in this Extended Abstract.

The algorithms will be compared as to the quality of the obtained results, complexity, and simplicity in implementation. In the discussion we will give our recommendations as to the heuristic that gives reasonably good results and at the same time works reasonably fast.

8 Conclusions

In contrast to the existing algorithms, our method optimizes the total word relatedness globally (within a relatively short text fragment) and not at each word independently. Namely, it looks for such a combination of senses that would optimize the total word relatedness. To find the global optimum, we have considered several heuristics to improve word sense disambiguation and gave the recommendations on the selection of the best one.

References

[1] Apresyan, Yu. D. New Comprehensive English-Russian Dictionary. Russky Yazyk, 1993.

[2] Araujo, L. Part-of-speech tagging with evolutionary algorithms. In A. Gelbukh (Ed.), Computational Linguistics and Intelligent Text Processing. CICLing-2002. Lecture Notes in Computer Science N 2276, Springer-Verlag, 2003, p. 230–239.

[3] Banerjee, S., and T. Pedersen. An adapted Lesk algorithm for word sense disambiguation using WordNet. In A. Gelbukh (Ed.), Computational Linguistics and Intelligent Text Processing. CICLing-2002. Lecture Notes in Computer Science N 2276, Springer-Verlag, 2003, p. 136–145.

[4] Edmonds, P., and A. Kilgarriff (Eds.), Journal of Natural Language Engineering, Vol. 9 no. 1, 2003. Special issue based on Senseval-2; www.senseval.org.

[5] Gelbukh, A., and G. Sidorov. Approach to construction of automatic morphological analysis systems for inflective languages with little effort. In A. Gelbukh (Ed.), Computational Linguistics and Intelligent Text Processing, CICLing-2003. Lecture Notes in Computer Science, N 2588, Springer-Verlag, 2003, p. 215–220.

[6] Alexander Gelbukh, Grigori Sidorov, San-Yong Han. *Evolutionary Approach to Natural Language Word Sense Disambiguation through Global Coherence Optimization*. WSEAS transactions, to appear.#

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

[8] Jiang, J.J. and D.W. Conrad. From object comparison to semantic similarity. In: *Pacling-99*, Pacific Association for Computational Linguistics, 1999, Waterloo, Canada, p. 256–263.

[9] Karov, Ya. and Sh. Edelman, Similarity-based word-sense disambiguation. *Computational linguistics*, Vol. 24, 1998, p. 41–59.

[10] Lawrence Davis, editor. Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York, New York, The 1991.

[11] Lesk, M., Automatic sense disambiguation using machine-readable dictionaries: how to tell a pine cone from an ice cream cone. Proc. of ACM SIGDOC Conference. Toronto, Canada, 1986, p. 24–26.

[12] Patwardhan, S., S. Banerjee, and T. Pedersen. Using measures of semantic relatedness for word sense disambiguation. In A. Gelbukh (Ed.), Computational Linguistics and Intelligent Text Processing, CICLing-2003. Lecture Notes in Computer Science, N 2588, Springer-Verlag, 2003, p. 241–257.

[13] Sidorov G. and A. Gelbukh, Word sense disambiguation in a Spanish explanatory dictionary. Proc. of TALN-2001, Tours, France, July 2–5, 2001, p. 398–402.

[14] Wilks, Y. and Stevenson, M., Combining weak knowledge sources for sense disambiguation. Proc. of IJCAI-99, 1999, p. 884–889