Evolutionary Approach to Natural Language Word Sense Disambiguation through Global Coherence Optimization

ALEXANDER GELBUKH,[1,2] 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

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*)." Current methods of such disambiguation involve local maximization of coherence: for every word they select the sense that has more in common with the surrounding words, taking into account all their senses. Since the words are processed independently, this is logically inconsistent: the choice of a sense for a word can be affected by the nearby words' senses other than the ones chosen by the algorithm when processing those words. This leads to sub-optimal coherence between the chosen senses of close words. In this paper, we consider global optimization of such coherence and show that it can be improved as compared with the best existing approaches, leading to superior results. Due to high dimensionality of the search space, a genetic algorithm is used to find a near-optimal combination of sense choices.

Key-Words: - Natural Language Processing, Word Sense Disambiguation, Lesk Relatedness Measure, Machine Translation, Information Retrieval, Genetic Algorithm.

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 a 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 of Michigan Lake, with an amazing view of Chicago's skyscrapers*

only the first one is relevant to the user's query, even if both contain the words *bank* and *Chicago*.

Another important application where automatic selection of the intended sense of a word is crucial is automatic translation. The two texts above are translated into Spanish as

– *Hay dos sucursales de Citybank en el área central de Chicago*,
– *El hotel está en la hermosa orilla del lago Michigan, con una impresionante vista a los rascacielos de Chicago*;

note different translations of *bank* (underlined). Obviously, incorrect choice of translation variants of a word renders the translated text completely incomprehensible. For example, a sentence *my cat broke when I used it to lift my car*, which has no reasonable interpretation, could be an (incorrect) translation of a Spanish phrase *mi gato se rompió 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 of 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 [7]. 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 suggest a new WSD technique, 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.

In the next section, we describe the related work in WSD and in application of genetic algorithms to disambiguation tasks. In Section 3 we present the Lesk word relatedness measure used to construct the fitness function for our genetic algorithm. In Section 4 we present the data structure we use to represent the relevant numerical information about the text, and in Section 5 we formally define the corresponding mathematical problem. In Section 6 we present our algorithm and in Section 7 describe the obtained results. In Section 8 we discuss these results and list some future work directions. Finally, in Section 9 the conclusions are drawn.

2 Related Work

In modern computational linguistics, there are two approaches to disambiguation. Perhaps the most popular one is based on statistics extracted, using machine-learning methods, from the texts themselves [15]. The source of information in this case are corpora, i.e., very large pools of texts (from several megabytes to several gigabytes), preferably with some kinds of linguistic markup. An extreme point of view within this approach is that it is enough to wait for the amount of available data to grow so that statistics would do the rest of the work [4]. There appears a problem of where these texts can be obtained and how they can be marked up automatically, semi-automatically, or manually.

Another approach is based on exploiting existing (and creating new ones) sources of linguistic knowledge, basically, different kinds of dictionaries [5]. Although not without its own problems, this approach allows to avoid some drawbacks of corpus-based approaches and to use available linguistic information in a reliable, ready-to-use form. In this paper, we follow this approach.

The work done in WSD can also be classified into two main groups: statistical methods [12], [15], [18], [19] and methods based on application of different knowledge sources [14], [16], [19], [20].

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 [10], [14]; the senses are chosen that are more probable in a given context. Padwardhan *et al.* [17] have compared different such measures and reported the Lesk relatedness measure [3], [14] 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 [13] 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 [17]. 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 [8].

For estimating relatedness between two sets of words, we use a measure analogous to the Dice coefficient [11], [12] 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 [19]. Note that we do not consider the synonyms as an additional “weak” knowledge source as in [20], 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 [14], 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

To represent 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.

– A sense s is characterized by a (variable dimension) matrix $M = M(w,s)$ of distances to the neighboring words.
– Its row i refers to a word $w_i \neq w$ from the same text fragment within a $W/2$ linear distance from the given word, where W is the window size. For the reasons discussed below, the number of rows in the matrix for different words can be different.
– The element j in the row i corresponds to the sense s_j of the neighboring word w_i. Note that the lengths of the rows vary within the matrix.
– The value $M_{ij}(w,s)$ specifies the relatedness (as discussed in Section 3) between the current sense s of the current word w and the sense s_j of the neighboring word w_i.

In our experiments we used a symmetric window of the same size W for all words, though this is not required by the algorithm. Since the words at the beginning and at the end of the text fragment have fewer neighbors, the matrix for these words has fewer rows. Some words—such as proper names—are absent from the dictionary and thus have no senses; in this case, the matrix M has no rows (if w has no senses) or zero length on the row (if w_i has no senses), correspondingly. Currently in our data structure we consider all words, including the functional words such as prepositions or articles and the words absent from the dictionary.

It is natural to expect that the relatedness measure be symmetric, i.e., $M_{ij}(w_k s_m) = M_{km}(w_i s_j)$. However, in practice it is not always so. This can be due to a number of reasons. For example, the formulas used in practice to calculate relatedness (see Section 3) are not symmetric with regards to the details of definition expansion or the use of synonyms and other related words. Also, the dictionaries used in this process are not always symmetric: for example, if the dictionary lists a word w as a synonym of a word u, it is not guaranteed that it also lists u as a synonym of w.

5 The Problem

The problem consists in selecting, for each word, one sense that is more likely to be the intended sense in 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 as 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 is 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.

One can attempt to solve this problem by a hypothesis that the correct sense for the word w_i is the one that is most related with the sense s under consideration. With this, instead of average as in the formula (1) above, the maximum relatedness in the row is to be considered:

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

Our experiments show that this option gives better results; see Section 7. However, it does not solve the problem: the sense selected by the max function in the formula and thus assumed by our hypothesis is not guaranteed to be chosen by the algorithm when processing the word w_i.

One can further play with the probabilities of the senses. For example, they can be automatically extracted from a corpus. The problem is that there are few available corpora marked with word senses, and they do not exist for many languages other than English. What can be done without sense-marked corpora is to assume that the less the number of the sense the higher its probability, i.e., that in the dictionary the lexicographers list the most probable senses first.

However, playing with probabilities does not resolve the logical inconsistency of the traditional method: the choice of the sense is affected by the other words' senses that are not guaranteed to be chosen in the final result.

A logically consistent problem formulation is as follows: choose such the combination of choices of 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.

6 The Algorithm

Since the task formulated in Section 4 is solved independently for each fragment, we consider only one fragment.

Let N be the number of words in the text fragment and n_i be the number of senses of the word w_i. Denote $\mathbf{N}$ the set of natural numbers. A combination of choices of senses is a function

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

such that $1 \leq f(i) \leq n_i$. Denote $\mathbf{F}$ the set of all such functions. An imaginable algorithm for solving the task is as follows:

```
for each sequence $f \in \mathbf{F}$
    for each word $w_k$
        $score(w_k) = \sum_i M_{i,f(w_i)}(w, f(k))$
    $score(f) = \sum_k^N score(w_k)$        (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

$$|\mathbf{F}| = \prod_i^N n_i \ .$$

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}.

To select the best sequence, we used a genetic algorithm. The parameters of the algorithm were as follows:

– The chromosome was a sequence of natural numbers varying from 1 to n_i, where n_i is the number of senses of the word w_i. If the word has no senses, i.e., is absent from the dictionary, the corresponding position in the chromosome was unused (in our implementation, for simplicity we just filled it with a value –1 in all individuals and do not change this value during mutation).

– The initial content of the pool was generated at random: for each individual and each position i in its chromosome, the value was generated randomly with the uniform distribution in the domain between 1 and n_i.

– The fitness function was $score(f)$ as defined by the formula (3) above. The objective was to maximize the fitness function.

– Generational type of genetic algorithm was used (as opposed to a steady-state type). That is, two pools were used, so that at each generation all parents were replaced with the respective offsprings, so that no individuals of a new generation could mate with individuals of the previous ones. This also means that the replacement method was set to appending: the new individuals are appended to the new pool.

– No generation gap was used, i.e., no predefined number of individuals was cloned to the new generation.

– The selection method was roulette wheel: the probability of an individual to be selected for crossover (or cloning, as described below) was proportional to its fitness value.

– The generation scheme was as follows: the selected pair of parents was replaced with two offsprings formed by exchanging the selected parts of the parents' chromosomes. With some probability the parents were simply cloned to the new generation instead of being mated.

– The crossover probability was determined by the parameter called crossover rate. It controlled the crossover option: whether the two selected individuals were mated and two children formed as a result of crossover or the two parents were simply copied to the new generation. The bigger crossover rate the more probably the parents are mated.

– The crossover method was simple: a single crossover point was selected at random; the genes up to and including the crossover point were copied to the respective child, and the remaining genes were copied to the alternate child.

– The mutation scheme was as follows: each child was selected or no for mutation with the prob-

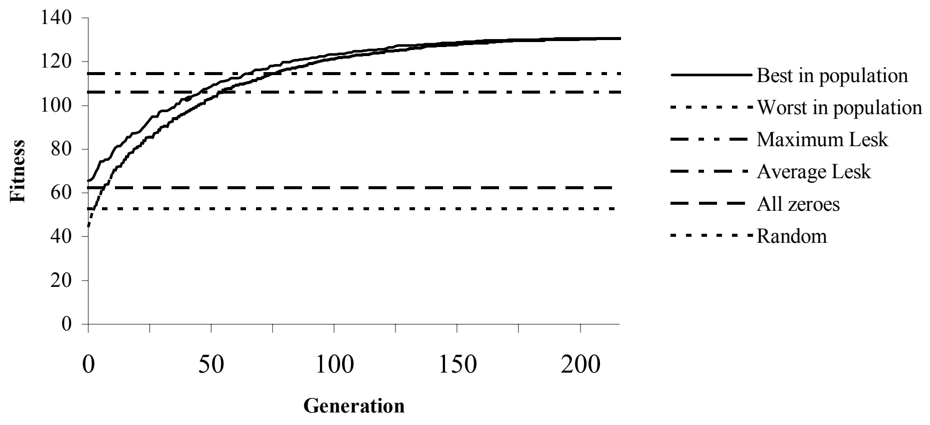


Fig. 1. Convergence of the algorithm.
The fitness of the best and worse individual in the population is shown for each generation. For comparison, the fitness of the solutions found with four baseline methods is shown.

ability determined by the parameter called mutation rate. If selected, a single mutation point *i* was selected at random (with the uniform distribution).

– A mutation at a point *i* was a random change of a gene in its respective domain, i.e., from 1 to n_i, where n_i is the number of senses of the word w_i.

– We used elitism to speed up convergence. This implies the following two modifications to the standard behavior of the algorithm. First, two copies of the best individual are cloned to the new generation's pool, thus ensuring its survival. Second, at each crossover action, out of four individuals—the two parents and two children—two best ones are placed into the new pool. In this way, if a child is not as good as either parent, it will not be selected, and a parent will survive instead.

– The termination condition was convergence: the algorithm stopped when all individuals in the pool had the same fitness value. In fact we experimented with continuing calculations after convergence and observed slight improvement in the results.

– We experimented with different pool sizes, mutation rates and crossover rates, as discussed in the next section.

7 Experimental Results

As baselines, we have implemented the following algorithms:

– *Random*. A random sense is selected for each ambiguous word; no heuristics applied.

– *All zeroes*. The first sense given in the dictionary is selected, regardless to the context. As our experiments show, this works better than a random choice, which can be explained by the fact that the lexicographers list the most frequent senses first.

– *Average Lesk*. The Lesk algorithm in the modification that maximizes the average relatedness of a sense to the context words, as in the formula (1) in Section 5.

– *Maximum Lesk*. The Lesk algorithm in the modification that maximizes the maximum relatedness of a sense to a context word, as in the formula (2) in Section 5.

We experimented with a 196 words long Spanish text taking from an Internet news site. In comparison with the baselines, our algorithm found better sense assignments, as can be seen from the following table, which shows the scores obtained for the corresponding assignments according to the formula (3) in Section 6; the same data can be seen in the graphical form in Fig. 1.

Algorithm	Score
Random	52.8
All zeroes	62.4
Average Lesk	106.1
Maximum Lesk	114.6
Genetic	**130.7**

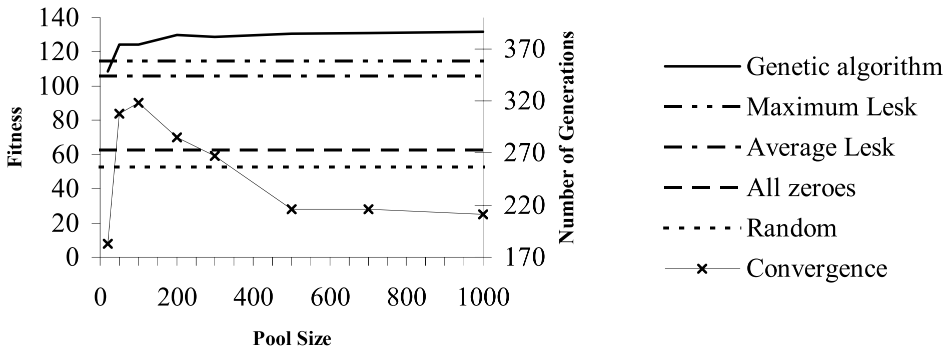
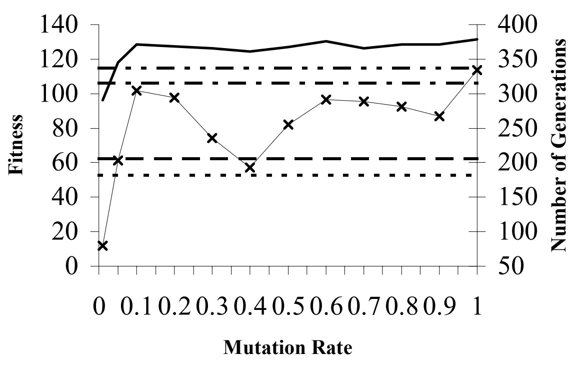


Fig. 2. Behavior of the algorithm depending on the pool size.
The fitness of the solution found after convergence is shown (left-hand axis) along with that of four baseline methods. The curve labeled "Convergence" shows the number of iterations until convergence (right-hand axis).



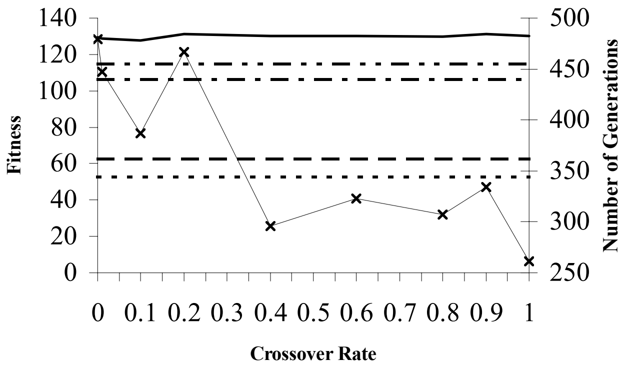


Fig. 3. Behavior of the algorithm depending on the mutation rate and crossover rate.
The curves have the same meaning as in Fig. 2.

In the table, the data for the genetic algorithm with the pool of 500 individuals and mutation rate and crossover rate both equal to 0.9 are given.

Fig. 1 shows the convergence curve for the algorithm with these parameters, the upper curve representing the best individual in the generation and the lower curve the worst one. The algorithm converged after 217 iterations, when all individuals reached the fitness of 130.651. The fitness reached by the baseline algorithms is shown for comparison; one can see that the algorithm surpassed the best baseline after 70 iterations. The total execution time to convergence was more than a minute on a Pentium III computer. We believe that this is explained by a very inefficiently implemented generic library we used; an efficient implementation should give much better timing.

We experimented with different parameters that can affect both convergence and the best score found. Fig. 2 shows how pool size affects the behavior of the algorithm; here mutation rate and crossover rate were set to 0.9. One can see that with more than 50 individuals our algorithm surpasses the best baseline, while with more than 200 individuals the results do not depend significantly on the pool size, showing very slight increasing as the pool size increases. However, after 500 individuals increasing the pool size does not lead to better convergence, while the increased amount of calculations deteriorates the speed of the algorithm (but not the result).

Fig. 3 shows that the algorithm is not very sensitive to the mutation rate and crossover rate. Still, best results were obtained with mutation rate 1.0 and crossover rate 0.9. In these calculations, the pool size was equal to 300. The mutation rate curve is given for crossover rate equal to 0.9 and the crossover rate curve for mutation rate equal to 1.0. On the other hand, convergence of the algorithm is significantly affected by the mutation rate and crossover rate, being the best values, with still reasonably good results, mutation rate of 0.4 and crossover rate of 1.0.

We also compared the automatically selected senses with the senses selected manually by a linguist prior to computational experiments. We tried different texts and different parameter settings for the genetic algorithms. In all our experiments, the assignment made by the genetic algorithm showed slightly better agreement with the manually assigned senses than all baselines except for the all zeroes heuristic.

Surprisingly, the latter heuristic performed slightly better than all versions of Lesk algorithm, including the genetic one. We explain this by low quality of text preprocessing tools (such as morphological normalizer) and of word relatedness measure we used. This circumstance, however, does not detract from the advantage of our algorithm over one of the state-of-the art techniques, but only shows that we should use better linguistic tools and/or annotation methodology.

8 Discussion and Future Work

Our experiments show that our method exhibits a superior results as compared with existing techniques. This is no surprise since it attempts to globally optimize the text cohesion, while the methods we used as baseline optimize it locally, i.e., independently for each word.

In the form we have presented it here, our method is good only for rather short texts. This is no problem in many cases, for example, when the method is applied to dictionary definitions [9] or short news reports. To apply our method to longer texts they should be partitioned into shorter fragments. This can be performed by linguistic-based text segmentation [6] or just using a kind of sliding window (intersecting segments). Probably a reasonable segmentation can improve the result. The choice of the segmentation method is a topic of the future research.

To improve the obtained results, we plan to try varying a number of other parameters of our algorithm. For example, the window size for non-zero relatedness measure (Section 3) should be more carefully selected; asymmetric or of variable-size windows can be tried.

Currently in our data structure we consider all words, including the functional words such as prepositions or articles and the words absent from the dictionary; we can try to eliminate them from the data structure. However, it is not quite clear what to do with words between functional and significant ones (e.g., Spanish word *sobre* has the meanings 'on', 'envelope', and 'to exceed').

Yarowsky's [21] principles "one sense per document" (or fragment) and "one sense per collocation" are to be implemented by their incorporation into the fitness function. Other similar heuristics can be incorporated, too: e.g., to give certain preference to the first sense, or to the most frequent sense, etc.

We plan to experiment with other word relatedness measures, for example, with those discussed in [17]. Also, we plan to try other global optimization methods, such as iterative backward-forward reestimation of the probabilities or dynamic programming.

Finally, we believe that the complexity of calculating the new individual's fitness after crossover, which in our current implementation is quadratic in the text fragment length (to calculate the new relatedness matrix), can be made linear: only the values near the crossover point need to be re-calculated. The implementation of this idea is not trivial and probably will require a more complicated representation of the chromosome, such a tree of smaller subfragments.

9 Conclusions

We have presented a novel algorithm for word sense disambiguation. The algorithm chooses the senses that optimize text cohesion in terms of a word relatedness measure (we experimented with Lesk measure). 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 used genetic algorithm.

Our experiments show that our method gives better results than existing state-of-the-art techniques.

Acknowledgements

The work was done under partial support of Mexican Government (CONACyT, SNI), IPN (CGPI, COFAA), and Invited Professorship Program of CAU. The first author is currently on Sabbatical leave at Chung-Ang University.

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] Brill, E. Processing natural language without natural language processing. In A. Gelbukh (Ed.), Computational Linguistics and Intelligent Text Processing, CICLing-2003. Lecture Notes in Computer Science, N 2588, Springer-Verlag, 2003, p. 360–369.

[5] Bolshakov, Igor A., Alexander F. Gelbukh, and Sofia N. Galicia-Haro. *Electronic Dictionaries: for both Humans and Computers*. In Václav Matoušek *et al*. (Eds.). Text, Speech and Dialogue, TSD-99. LNCS, N 1692, Springer-Verlag, 1999, p. 358–361.

[6] Bolshakov, I.A., and A. Gelbukh. Text segmentation into paragraphs based on local text cohesion. Text, Speech and Dialogue, TSD-2001. Lecture Notes in Artificial Intelligence N 2166, Springer-Verlag, 2001, pp. 158–166.

[7] 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.

[8] 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.

[9] Gelbukh, A., and G. Sidorov. Automatic selection of defining vocabulary in an explanatory dictionary. In A. Gelbukh (Ed.), Computational Linguistics and Intelligent Text Processing, CICLing-2002. Lecture Notes in Computer Science N 2276, Springer-Verlag, 2003, p. 300–303.

[10] 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.

[11] 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.

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

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

[14] 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.

[15] Manning, C. D. and H. Shutze, Foundations of statistical natural language processing. Cambridge, MA, The MIT press, 1999, 680 pp.

[16] McRoy, S., Using multiple knowledge sources for word sense disambiguation. *Computational Linguistics*, Vol. 18(1), 1992, p. 1–30.

[17] 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.

[18] Pedersen, T., A baseline methodology for word sense disambiguation. In A. Gelbukh (Ed.), Computational Linguistics and Intelligent Text Processing, CICLing-2002. Lecture Notes in Computer Science N 2276, Springer-Verlag, 2003, p. 126–135.

[19] 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.

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

[21] Yarowksy, D., Word-sense disambiguation using statistical models of Roget's categories trained on large corpora. Proc. of COLING-92, Nante, France, 1992, p. 454–460.