Global Journal of Technology & Optimization

Abualhaija and Zimmermann, Global J Technol Optim 2016, 7:1
http://dx.doi.org/10.4172/2229-8711.1000193

Research Article Open Access

Solving Specific Domain Word Sense Disambiguation using the D-Bees Algorithm

Sallam Abualhaija* and Karl-Heinz Zimmermann

Institute of Embedded Systems, Hamburg University of Technology, Schwarzenberg (Campus E), D-21071 Hamburg, Germany

Abstract

Word sense disambiguation is the problem of finding the most likely senses for a sequence of words in a given context. Disambiguation is a major step in most of the text applications. However, the meanings of the words are highly dependent on the domain of the text. Recently, word sense disambiguation is being addressed as an optimization problem. For this, metaheuristics like simulated annealing and D-Bees are developed. In this paper, we try to answer the question about the compatibility of general domain algorithms to solve specific domain word sense disambiguation. For this, we propose two variants of the D-Bees algorithm to include the domain information into the disambiguation process. The concepts proposed in this paper are general and can be adapted to other algorithms. It will be concluded that the D-Bees algorithm is suitable for solving specific domain word sense disambiguation. It has a robust performance in general and achieves competitive results compared with the simulated annealing method for different datasets.

Keywords: Specific domain; Disambiguation; Algorithm; D-Bees

Introduction

Word Sense Disambiguation (WSD) is the problem of identifying the most likely sense of a target word, or a sequence of words, in a given context [1]. For example, the word mouse has two different meanings in the following two sentences: "The mouse eats cheese" and "I want to buy a new mouse for my computer". The word mouse in the previous two sentences refers to "a rodent" and "a computer device" respectively.

WSD is not a stand-alone problem [2,3] rather it is used implicitly in many applications like machine translation [4] information retrieval [5] lexical simplification [6] and others.

Disambiguating words in a particular domain is referred to as specific domain WSD. Considering the domain of the text in WSD is important in particular applications like machine translation in order to achieve high accuracy. For example, the word bug should be translated to the German word Wanze if it means "an insect" in some biological context. If it has the computing sense "a fault or defect in a computer program", then it should be translated into German as Fehler or Fehlfunktion.

There are several methods to solve the WSD problem for general domain that are usually divided into three interrelated categories: supervised, unsupervised, and knowledge-based methods [2,1].

Supervised methods use semantically annotated corpora. In these methods, a classifier is trained using a subset of the annotated corpora (training set), then evaluated using the another subset of the annotated corpora (test set). The classifier should be able to classify new examples correctly [1].

Unsupervised methods use unannotated corpora. They are mainly based on the assumption that words which occur in similar contexts are likely to have similar meanings. The result is a set of clusters of related words, each of which conveying a particular sense [7].

Finally, knowledge-based methods rely on machine readable dictionaries such as WordNet [8]. These approaches are applicable to any text, as long as the words are covered by the used dictionaries.

The Lesk algorithm [9] is a well-known knowledge-based method that disambiguates two words by calculating the contextual overlap between these definitions of their senses. For example, each of the words pine and cone has a sense that includes the terms evergreen tree in their sense definitions, so these two senses will be assumed to disambiguate each other in case pine and cone co-occur in the same context.

This algorithm is intuitive and simple to implement. However, it fails when there is no overlap found between the sense definitions. Therefore, variants of the Lesk algorithm are proposed to augment the definition of a word meaning with definitions of semantically related words [10].

Further researches are being done on the knowledge-based methods although the supervised methods achieve better results [11]. The reason is that the process of annotating corpora needs strenuous effort and has to be repeated for every language and for various domains.

Moreover, the same language evolves by time which means even more effort to get new examples if new terms appear; e.g., the word tweet nowadays means "the sound of a bird" or "an entry posted on Twitter" [12]. Therefore, the coverage of supervised methods is limited by the available set of annotated examples.

Recently, WSD is being addressed as an optimization problem by applying the Lesk algorithm to a a text larger than two words [13]. The straight forward method is to consider all possible definition combinations. This may lead to combinatorial explosion, that is the

***Corresponding author:** Sallam Abualhaija, Institute of Embedded Systems, Hamburg University of Technology, Schwarzenberg (Campus E), D-21071 Hamburg, Germany, Tel: +49 (0) 40 42878-3255; Fax: +49 (0) 40 42878-2798 E-mail: sallam.abualhaija@tu-harburg.de

Received April 25, 2016; **Accepted** May 02, 2016; **Published** May 06, 2016

Citation: Abualhaija S, Zimmermann KH (2016) Solving Specific Domain Word Sense Disambiguation using the D-Bees Algorithm. Global J Technol Optim 7: 193. doi:10.4172/2229-8711.1000193

time complexity grows exponentially with the problem size and so the problem becomes unsolvable within reasonable time [14].

In the previous example: "The cat chases the mouse in the yard", the sentence has three open-class words each of which having several possible meanings as given by WordNet lexicon [8]: cat(8), chase(4), mouse(4), yard(9) resulting in a total of 1,152 sense combinations for this small sentence. This shows that applying the straight forward method on a simple sentence might lead to the problem of combinatorial explosion.

Metaheuristics like simulated annealing [15] and D-Bees [16] are global algorithms that can be used to avoid the combinatorial explosion problem.

Specific domain WSD introduces new challenges. The main questions that arise here are to what extent can the general domain WSD systems still be used within the specific domain application? Also how should they be adapted to fit this problem if their performance is not sufficient?

In this paper, we propose applying the D-Bees algorithm on WSD problem for specific domain and try to answer the previous questions. The experiments are conducted on two different datasets. The same concepts are performed using the simulated annealing algorithm and the results of both algorithms are compared and analyzed thoroughly. The major contribution of this paper is proposing two general methods to include the information from the domain into the WSD process. This is applied on the D-Bees and simulated annealing here, but can be applied to other algorithms as well.

The remainder of this paper is organized as follows. Section 2 gives an overview of the related background and describes the used algorithms in brief. Section 3 explains adjusting the D-Bees algorithm to solve the WSD problem for specific domain. For this, we introduce the domain detection module and two different variants of the D-Bees algorithm. Section 4 describes the results and experiments in details. At the end of the section an analysis is provided about the performance of the D-Bees algorithm in comparison with simulated annealing. The conclusion is then given in section 5.

Background

The D-bees algorithm for general domain

Inspired by [13], the WSD optimization problem can be defined as an optimization problem to disambiguate a sequence of words simultaneously [16].

Definition 1 Let $W=(w_1,w_2,\ldots,w_n)$ be a sequence of n words to be disambiguated and a sequence of senses $\sigma=(s_1,s_2,\ldots,s_n)$ be the corresponding sense s_i, $1\le i\le n$ for each word Wi. Let $\mathcal{S}=\{\sigma_1,\ldots,\sigma_m\}$ be the set of all sequences of senses that represent sense combinations of the words in the sequence w. Then the objective function becomes

$$argmax_{\sigma\in\mathcal{S}}\ell(\sigma), \tag{1}$$

where ℓ is the score assigned to a sequence of senses based on the semantic relatedness. The score is calculated using a variant of Lesk (eLesk) as in [10].

The D-Bees algorithm aims at solving the WSD as an optimization problem and it can be summarized as follows [16]. At first, one word is chosen from the target words to represent the hive which produces bee agents and sends them to other words in the context window.

The number of bee agents equals the number of senses of the hive. In this way, each bee agent holds one sense in its memory and searches for the senses that have a higher similarity value with the sense it holds.

Initially, the path contains a sense of the target word from which the bee has created and the quality is set to zero. After each step, the bee agents update their local memory. They append the chosen sense to the path and update its quality by adding the similarity value incrementally.

The bee agent moves a step further until the number of moves is reached and by this it accomplishes a forward pass. Then it initiates the backward pass by returning to the hive holding a partial solution. In the hive, the agents exchange information on a virtual dancing floor.

During the backward pass, the bee agents with good partial solution advertise their solutions by performing a waggle dance. Each bee calculates the loyalty probability, based on which the bee agent decides whether to stay loyal to its path or to become uncommitted and follow one of the advertised solutions. It follows that becoming a recruiter depends mainly on the goodness of the partial solution found so far.

The forward and backward passes are alternated until there are no more target words to be disambiguated. The bee agent with the best found solution in terms of path quality is stored. Finally, the best solution is returned as the output of the disambiguating process.

Simulated annealing for general domain

Simulated Annealing (SA) is based on the metal annealing process. The metal is heated and then cooled slowly to obtain a stable state [17].

SA has been applied to the WSD problem by Cowie et al. [15]. Given a sequence of n words $W=(w_1,\ldots,w_n)$ the SA algorithm starts with an initial sequence (σ) by assigning a sense to each word in the sentence, randomly or based on some predefined criterion.

The algorithm searches for a sequence of senses that has the highest number of overlapping words among the definitions. This overlap is called redundancy (R). To do this, first all the definitions of the senses are retrieved, preprocessed and stored in a list. For this list, a frequency map is created, where each word has a matching frequency value measuring how often the word occurs in the list. The redundancy is then the summation of the frequency values minus one over all the list such that the words that occur once do not influence the redundancy value.

WSD in this case can be defined as a minimization problem, where each sequence has an associated energy value (E) and the goal is to find the sequence (σ) with the minimum energy value. The energy function E represents the dissimilarity among words and can defined as follows:

$$E=\frac{1}{1+R}, \tag{2}$$

where R is the redundancy.

A random modification is done on the initial solution, by changing a sense of a randomly chosen word. The energy value is calculated for and a decision whether to accept the new sequence or not is made. SA accepts a "not-very-good" solution based on the rejection probability in order to escape from the local optima hoping that a better solution will be found in the later iterations.

This process is repeated until the maximum number of iterations is reached, or the best solution found so far cannot be changed significantly. Finally, the best found solution is returned.

Related work

The domain is included in the dictionaries under the name of subject code [1]. Subject codes are used to define which senses belong to which domains; e.g., the word bass expresses different meanings based on the domain in which it is used, in music it means "the lowest part of the musical range" and in biology it means "a type of fish". Subject codes can be used to detect the domain of a certain text by counting their frequency overall words in the text. This is beneficial for the WSD specific domain system by giving a preference to the senses that share the same subject code.

The domain information are included in WordNet [8]. Synsets have at least one domain label [1]. Domains can include synsets of different syntactic categories, and they may group multiple different senses of a certain word into a semantic cluster. This decreases the ambiguity level in the case of specific domain disambiguation.

Magnini et al. [18] have presented a system to solve the specific domain WSD problem under the hypothesis, called DDD (Domain Driven Disambiguation). This hypothesis indicates that it is possible to establish associations among words in a coherent text using their domain labels, such that the related senses maximize the domain similarity.

To this end, given a target word w_1 and a context window $W = (w_2, \ldots, w_n)$, the disambiguation process can be done in three steps [1]:

Create the domain vector of the context window $T(W) = \cup_{i=2}^{n} T(w_i)$ (± 50 words around the target word were used in the DDD system).

Find the domain vector for each sense s_j of the target word $T(s_{1j}) = (t_{j1}, \ldots, t_{jk}), 1 \le j \le m_1, k \ge 1$, such that m_1 is the number of senses of the target word, and k is the number of domain words found for this sense.

Choose the sense s^* of the target word such that

$$s^* = argmax_{j=1}^{m_1} \ell(T(W), T(s_{1j})). \quad (3)$$

where ℓ is the semantic similarity calculated between the sequence of the domain words of the context window and the target word.

The domain vectors can be created using the associations between the words synsets and the domain label provided in WordNet. Then, the sense that maximizes the similarity with the relevant domain of the context is selected. The DDD system was tested in Senseval 2 [19] and achieved a precision of 75% and a recall of 36%. The reason of achieving a low recall value is that only a subset of the word senses in a document are actually related to the domain of the context.

The D-Bees Algorithm for Specific Domain

Adapting the D-Bees algorithm from the general domain to a specific one starts by feeding the algorithm with domain information about the dataset before the disambiguation process takes place. Therefore, the first step is detecting the domain automatically. This is possible by using the suitable semantic relations provided by WordNet to indicate the domain.

Before starting the disambiguation process, the domain detection procedure is performed on the whole dataset. Given a text to be disambiguated $W = (w_1, w_2, \ldots, w_n)$, detecting the domain is done as follows:

1. Retrieve the sequence of possible synsets $S = (s_{i1}, s_{i2}, \ldots, s_{im_i})$ from Word Net for each word *wi*, where $m_i \ge$ is the number of senses, $1 \le i \le n$.

2. For each synset s_{ij}, get the semantic related synsets corresponding to the semantic relation category.

3. Create a frequency table of two columns. In the first column, store the domain words retrieved for all synsets of the target words in the text. In the second column, store how many times this domain word is associated with the synsets in the whole text.

4. Sort the frequency table in descending order, such that the first entry represents the most frequent domain word in the whole text.

5. The idea is then to work with only the top-*k* domains, we used the top-3 domains in our experiments.

We have included the domain words in the disambiguation module of the D-Bees algorithm using two different approaches.

D-Bees variant 1: Domain information in choosing next synset

The D-Bees algorithm for general domain is discussed earlier. Given a sequence of words to be disambiguated $W = (w_1, w_2, \ldots, w_n)$, the bee agents explore the senses of the words and append each time a sense to the build up an admissible solution. Choosing the next sense can be done either based on the usage frequency or uniformly at random.

In the specific domain D-Bees variant 1, we incorporate the domain words by choosing the next sense during the disambiguation process. This variant is motivated by the fact that the distributions and predominant senses vary in a specific domain text [20]. Therefore relying on on frequency of usage to choose the next senses might not be result in "good-enough" solutions.

The alternative is to choose the next sense that has the highest overlap with the top-*k* domain words. Let $W = (w_1, w_2, \ldots, w_n)$ be the sequence of words to be disambiguated. The domain words can be defined as $T(W) = \cup_{i=1}^{n} T(w_i)$. Let $T = (t_1, t_2, \ldots, t_k)$ be the top-*k* frequent domain words in the whole text (or dataset) sorted in descending order. Choosing the next sense is then performed by selecting the sense which has maximum number of overlapping domain words with the text. Let $\sigma(w_i)$ be the selected next sense of the words w_i. It can be defined formally as follows:

$$\sigma(w_i) = argmax_{j=1}^{m_i} \{| \text{overlap}(T, T(s_{ij})) |\} \quad (4)$$

where $T(s_{ij})$ is the domain words obtained from the j^{th} sense of the word w_i, $1 \le i \le n$.

The D-Bees variant 1 works as follows:

Detect the domain for the whole dataset before starting the disambiguation.

1. The disambiguation component works per sentence. Similar to the generic D-Bees algorithm, after choosing the hive and initializing the bee agents are supposed to choose a next sense to append it to the partial solution they found so far.

The next word *wi* has the sequence of senses $S(w_i) = (s_{i1}, s_{i2}, \ldots, s_{im_i})$.

Retrieve the semantic relations for each sense s_{ij} from WordNet, namely category. The result is sequence of domain words associated with this sense $T(s_{ij})$.

Choose the next sense from the sequence of senses as in Eq. 4.

If no overlapping domain words can be found between T and $T(s_{ij})$, then choose the next sense uniformly at random or based on the usage frequency as in the generic D-Bees.

The forward and backward passes as well as the loyalty and recruitment process are performed similar to the generic D-Bees algorithm.

The general flow chart that describes the D-Bees algorithm variant 1 for specific domain WSD is illustrated in Figure 1.

D-Bees variant 2: Domain information per sentence

The second approach is to append the top-k domain words to each sentence before disambiguating it, and then remove them after the disambiguation is completed as illustrated in Figure 2. In this case, the domain words are participating in the disambiguation process. The definitions of the domain words are influencing the objective function, since we use the eLesk similarity measure.

This variant enriches the sense definitions with the domain information. Therefore the result of the disambiguation process should be more domain-relevant senses. In this way, the same generic D-Bees algorithm can be used with some minor changes. Besides the domain detection module, a component is created that appends the domain words and the removes them to and from each sentence.

In more details the D-Bees algorithm variant 2 works as follows:

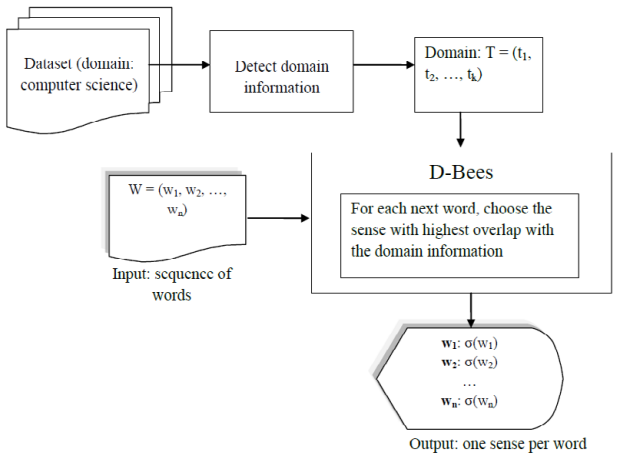

Figure 1: D-Bees variant 1 for the specific domain WSD.

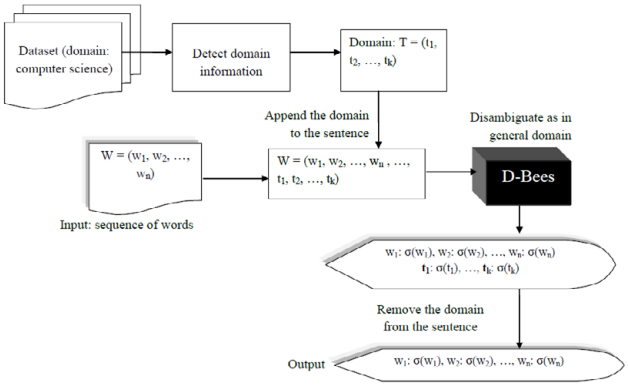

Figure 2: D-Bees variant 2 for specific domain WSD.

• Detect the domain words of the text before the disambiguation process same as in variant 1. Let $T=(t_1,t_2,\ldots,t_k)$ be the domain words sorted in descending order based on the frequency of occurrence in the text.

• Let $W=(w_1,w_2,\ldots,w_n)$ be the sequence of words to be disambiguated. Then append the domain words to the sequence of the words:

$$W_T = append\ (W,T)=(w_1,w_2,\ldots,w_n,t_1,t_2,\ldots,t_k)\,.$$

• Perform the D-Bees disambiguation procedure on the W_T the same way as for the general domain. This means that the domain words are considered like the target words and will be disambiguated.

The domain words might also be polysemous but they are definitely more specific than the target words; e.g. the domain music and the target word bass. The D-Bees algorithm benefits from the advantage of considering the most frequent senses during the disambiguation; e.g., the case of music.

• Delete the domain words from the sentence: W=*delete* (W,T) such that only the words in the original sequence will be returned as the output of the disambiguation process.

The words that are not specific domain do not have a significant influence on the domain detection. This fact makes this variant more appealing because the D-Bees algorithm focuses on the specific domain senses but still handles the target words well from a general domain perspective.

Results and Analysis

The dataset

The specific domain word sense disambiguation is introduced in [20] task 17: all-words WSD on a specific domain for different languages: Chinese, Dutch, English and Italian [20]. The dataset provided with the task includes a test set and background texts, both of which focus on the environmental domain. Table 1 summarizes the statistical information about the test sets.

The test set for the English language contains three texts focusing on the environmental domain divided between 1,032 nouns and 366 verbs. The target words are annotated by only one expert.

The dataset includes three texts compiled by the European Center for Nature Conservation and Worldwide Wildlife Forum [20]. Only nouns and verbs are tagged as being target words and should be disambiguated. The inter-annotator agreement is not applicable for the English dataset because it was annotated by only one expert.

The first sense baseline has scored a precision of 50.5% on the English test set, where the most frequent senses are taken from the WordNet. The random baseline has scored a precision of 23.3% which indicates a high average of polysemy in comparison with other languages: 32.1%,32.8%,29.4% on Chinese, Dutch and Italian, respectively [20].

In addition to this dataset, we have conducted experiments on the

Language	Total	Noun	Verb	IAA
Chinese	3989	754	450	0.96
Dutch	8157	997	635	0.90
English	5342	1032	366	n/a
Italian	8560	1340	513	0.72

Table 1: Dataset statistics.

fourth text (d004) in SemEval 2007 task 7 [11]. The lemmas and part-of-speech are provided for the target words. The text has 677 different instances of nouns, verbs, adjectives and adverbs. The domain of the text is computer science. Based on our experiments on d004, the precision of the random sense baseline is 60.71%, while for the first sense 75.18%.

To evaluate any WSD system, the following metrics are used. First, coverage describes how many target words are disambiguated by the system out of the total target words. Precision is the number of target words that are correctly disambiguated divided by the total number of target words that the system disambiguates and recall is the number of correctly disambiguated target words divided by the total number of target words in the dataset. Finally, the F-measure is the harmonic mean of the precision and recall values.

Domain detection

In the [20] corpus the target words are tagged, but neither their part of speech nor lemmas are provided. Therefore the performance of the WSD system is partially influenced by the POS tagger as well as the lemmatizer. The coverage is $^{100\%}$ for all algorithms; i.e., precision = recall = F-measure. Thus we will report the precision only.

We have used two approaches in order to detect the domain of a given text. The first approach (all texts) considers the domain words over all the texts in [20] task 17 corpus.

We have developed the second approach (per text) because the corpus consists of three different texts. The second approach considers the domain words for each text in the corpus and apply it sentence wise within the same text. This improves the results because then we limit the range of domain detection into one coherent text. The detected domain words are then more precise in the second approach. The results are obtained by applying the D-Bees variant 1 discussed earlier. Table 2 summarizes the results.

In Figure 3, we compare the performance of the domain detection approaches on [20] task 17 corpus divided per text. It can be observed that for all texts detecting the domain within one text outperforms

Detection-Method	Precision (%)
All Texts	44.56
Per Text	47.28

Table 2: Domain detection.

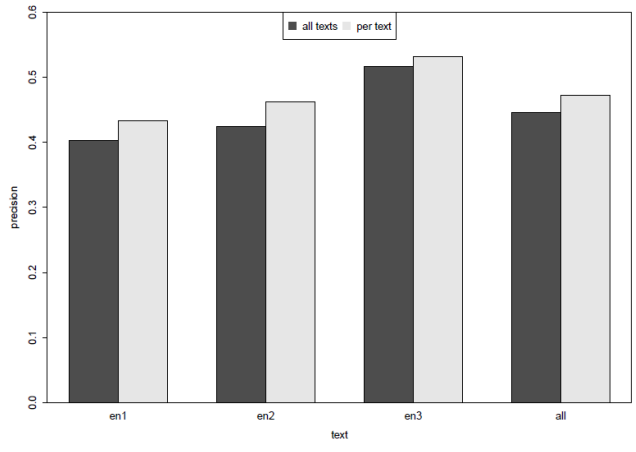

Figure 3: A comparison between the domain detection approaches.

incorporating the domain information from the whole corpus.

The top two domain words in each text found by the domain detection module for both corpora are listed in Table 3.

It can be observed that the dominant domain words in the text d004 are precise and more specific. Therefore, choosing the senses which have the maximum overlap with the domain words should lead to a good performance. This argument has been proved by the results of the D-Bees algorithm. The domain words have a remarkably high frequency value (91 each) in comparison with the texts of [20] texts. This means that they are shared among many words senses in the text.

On the other hand, in [20] the top domain words have less frequencies. This means when selecting the next sense of a particular word, it is less likely to find one with an overlap with the domain words. However including the domain words in the disambiguation process for each sentence enforces a bias towards the domain relevant senses in the text. It can also be that the domain of environment is broader than the computer science. Therefore, the domain words in the environmental field have a larger polysemy average and need to be disambiguated in comparison with that in the computer science field.

It can be assumed that the domain words can be highly affected by the named entities in the text as shown in Table 3. There might be some instance of rivers in the second text en2 that lead to the domain word river; e.g., Jordan River. It might be useful to include some key words manually to the whole text based on the domain. However, to avoid the necessity of manual intervention, the algorithm can consider more domain words than the top-3 as in our experiments. This ensures that the existence of many named entities will not mislead the search process.

Comparison with simulated annealing

We have developed the same disambiguation approaches using the Simulated Annealing (SA) algorithm for the specific domain WSD (more about this can be found in [21]).

SA is a metaheuristic that was adapted to solve the WSD optimization problem. In SA, the disambiguation process starts with an initial sequence of senses corresponds to the sequence of words to be disambiguated. SA variant 1 includes the domain information from each text in the corpus for modifying the initial sequence randomly. To do this, first a random word is chosen for which the sense will be changed. Then a sense is chosen for this word according to variant 1 explained earlier. SA variant 2 works similarly as D-Bees variant 2 by adding the domain words to the sentence.

A comparison is given in Table 4 between the D-Bees and the simulated annealing algorithms including the baselines for both corpora. It can be observed that the D-Bees variant 1 performs better than variant 2 on the text d004 unlike SA. This also contradicts the

Text		Domain	Frequency
6*[20]	2*en1	Narcotic	28
		Biology	27
	2*en2	River	10
		Educational Activity	9
	2*en3	Biology	48
		Biological Science	48
2*[11]	2*d004	Computer Science	91
		Computing	91

Table 3: The top-2 domain words per text.

results on [20]. The reason might be that d004 is a coherent text in the domain of computer science.

Figures 4 and 5 visualize the results. It is worth mentioning that the results on d004 are obtained using the best parameters found by the ILS parameter estimation method trained on a subset of the [20] corpus.

The results of D-Bees in comparison to SA divided by the text and POS are visualized in Figure 6.

Algorithm		Precision (%)	
		[20] task 17	[11] (d004)
First Sense		50.5	75.18
3*D-Bees	Generic	48.93	76.07
	Variant 1	47.28	81.5
	Variant 2	50.00	75.33
3*SA	Generic	24	68
	Variant 1	25	69
	Variant 2	40	71
Random Sense		23	60.71

Table 4: A comparison between the D-Bees and SA algorithms for specific domain.

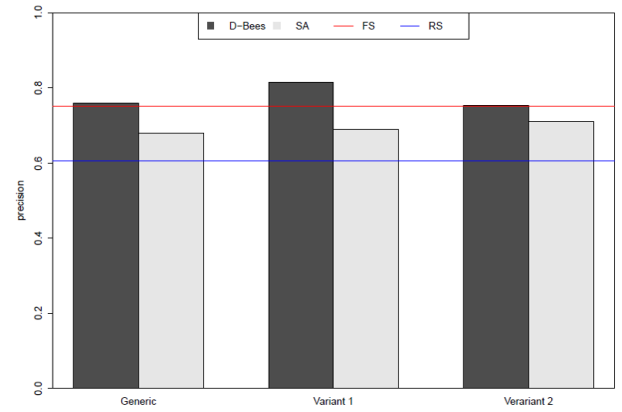

Figure 4: A comparison between the D-Bees and SA algorithms on SemEval 2007: d004.

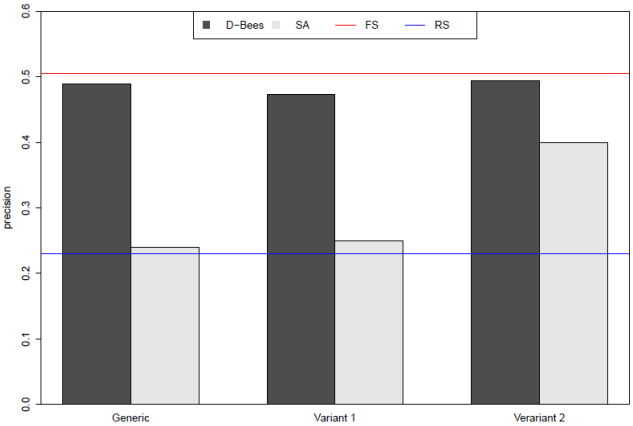

Figure 5: A comparison between the D-Bees and SA algorithms on SemEval 2010.

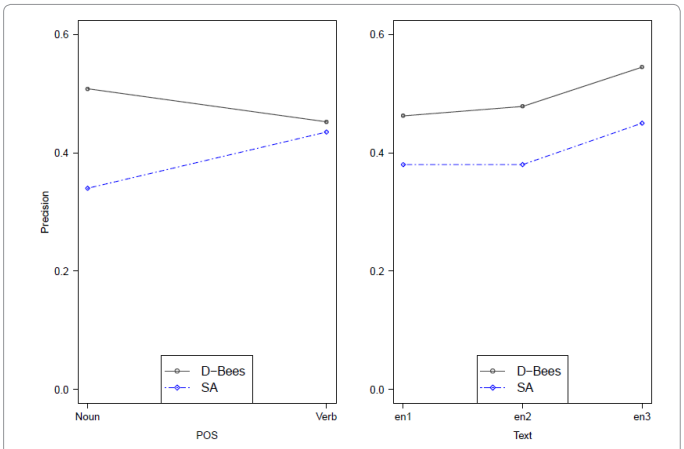

Figure 6: The results on specific domain per text and POS.

Comparison with other participating systems in [20] task 17

Eleven systems have participated in this task and submitted more than thirty runs. The systems are classified in three categories based on the way they trained or obtained the senses of the target words, supervised, weakly supervised or knowledge-based systems. Below we describe the systems with which we compare the D-Bees results [20]:

Treematch uses a knowledge-based method that requires a dictionary and untagged text as an input.

Kyoto represents a free reimplementation of the WSD method presented in [22].

CFILT is a specific domain knowledge-based system. The system works by first identifying the domain dependent words using the background dataset. Then the system selects the most representative synsets within the domain using a graph based on the hyponyms in WordNet and a breadth first search method. They have added manually disambiguated examples from the domain as seeds. Their best run is under the weakly supervised category and it was ranked first.

IIITTH is based on the personalized Page Rank algorithm over a graph constructed from WordNet similar to [22].

RACAI uses a mapping to the domains provided by WordNet in order to limit the domains of the words in the test set.

HIT-CIR estimates the predominant sense from the raw test. This is done by calculating the frequency information in the background text.

• WS acquires information by querying the local context of a given target word in the Web.

The sense is chosen based on the relatedness between the senses of the target word and the information obtained from the Web.

Besides the first sense baseline, we compare the results of the D-Bees algorithm variant 2 only with the participating knowledge-based systems. The results in Table 5 are sorted in descending order based on the recall values.

It can be observed that the D-Bees algorithm performs on par with the top ranked knowledge-based systems, although both variants of the D-Bees algorithm do not consider the background text in the disambiguation process.

System	Precision	Recall	R nouns	R verbs
FS Baseline	0.505	0.505	0.519	0.464
D-Bees	0.500	0.500	0.5141	0.4605
CFILT-3	0.512	0.495	0.516	0.434
Treematch-1	0.506	0.493	0.516	0.426
Treematch-2	0.504	0.491	0.515	0.425
kyoto-2	0.481	0.481	0.487	0.462
Treematch-3	0.492	0.479	0.494	0.434
RACAI-MFS	0.461	0.460	0.458	0.464
UCF-WS	0.447	0.441	0.440	0.445
HIT-CIR-DMFS-1.ans	0.436	0.435	0.428	0.454
UCF-WS-domain	0.440	0.434	0.434	0.434
IIITH2-d.r.l.baseline.05	0.496	0.433	0.452	0.39

Table 5: A comparison between the D-Bees algorithm and the knowledge-based systems on [20] Task 17.

The D-Bees algorithm is adapted to the specific domain WSD in a way similar to the DDD method described earlier [18]. However unlike the DDD method, it scores a stable recall value. The reason is that including the domain words in the disambiguation process favors the domain relevant senses. This can be achieved in the D-Bees algorithm because the semantic similarity function considers the sequence of words at once.

During the disambiguation the definitions of the domain words are included each time. Choosing next senses based on their usage frequency is highly beneficial to disambiguate the domain labels that lead the search to focus on the senses related to the domain.

Conclusion

In conclusion, the performance of the D-Bees algorithm is robust to some extent and the algorithm can be adapted to the WSD for specific domain with minor modifications. Based on the conducted experiments, it can be inferred that the D-Bees algorithm performs better than the simulated annealing.

Adjusting a WSD algorithm from general to specific domain starts by detecting the domain in the dataset. Then the two variants that are presented in this paper can be used to adapt any algorithm from general to specific domain.

The D-Bees is comparable to the first sense baseline. However the D-Bees algorithm takes the context into consideration and gives a priority to the more frequent senses. Therefore, even if they perform on par, the D-Bees algorithm can be improved unlike the first sense baseline. As a future enhancement, the D-Bees algorithm should consider the background texts to find more accurate domain words. It is also possible to include some domain words manually once for the whole dataset in order to cause a bias in sense selection.

References

1. Agirre E, Edmonds PG (2007) Word sense disambiguation: Algorithms and applications.Springer Science & Business Media volume: 33.
2. Edmonds P (2005) Disambiguation, lexical. Encyclopedia of Language and Linguistics. (2ndedn), Elsevier.
3. Nancy I, Véronis J (1998) Introduction to the special issue on word sense disambiguation: the state of the art.Computational linguistics 24: 2-40.
4. Vickrey D, Biewald L, Teyssier M, Koller D (2005) Word-sense disambiguation for machine translation. In Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing. Association for Computational Linguistics.
5. Sanderson M (1994) Word sense disambiguation and information retrieval. In Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval. Springer-Verlag New York Inc PP: 142-151.
6. Specia L, Jauhar SK, Mihalcea R (2012) Semeval-2012 task 1: English lexical simplification. In Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation. Association for Computational Linguistics PP: 347-355.
7. Schütze H (1998) Automatic word sense discrimination.Computational linguistics 24: 97-123.
8. Miller GA, Beckwith R, Fellbaum C, Gross D, Miller KJ (1990) Introduction to wordnet: An on-line lexical database*. International journal of lexicography 3: 235-244.
9. Lesk M (1986) Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In Proceedings of the 5th annual international conference on Systems documentation, ACM PP: 24-26.
10. Banerjee S, Pedersen T (2002) An adapted lesk algorithm for word sense disambiguation using wordnet. In Computational linguistics and intelligent text processing,Springer pp: 136-145.
11. Navigli R, Litkowski KC, Hargraves O (2007) Semeval-2007 task 07: Coarse-grained english all-words task. In Proceedings of the 4th International Workshop on Semantic Evaluations. Association for Computational Linguistics pp: 30-35.
12. Tahmasebi N, Risse T, Dietze S (2011) Towards automatic language evolution tracking, a study on word sense tracking. In Joint Workshop on Knowledge Evolution and Ontology Dynamics.784.
13. Pedersen T, Banerjee S, Patwardhan S (2005) Maximizing semantic relatedness to perform word sense disambiguation.University of Minnesota supercomputing institute research report UMSI 25: 2005.
14. Dorigo M, Stützle T (2004) Ant colony optimization. Bradford Company, Scituate, MA, USA.
15. Cowie J, Guthrie J, Guthrie L (1992) Lexical disambiguation using simulated annealing. In Proceedings of the 14th conference on Computational linguistics-Association for Computational Linguistics 1: 359-365.
16. Abualhaija S, Zimmermann KM (2016) D-bees: A novel method inspired by bee colony optimization for solving word sense disambiguation. Swarm and Evolutionary Computation 27: 188-195.
17. Dréo J, Siarr P, Pétrowski A, Taillard E (2006) Metaheuristics for hard optimization. Springer-Verlag.
18. Magnini B, Strapparava C, Pezzulo G, Gliozzo A (2002) The role of domain information in word sense disambiguation.Natural Language Engineering 8: 359-373.
19. Edmonds P, Cotton S (2001) Senseval-2: Overview. In the Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems, SENSEVAL '01, Stroudsburg, PA, USA. Association for Computational Linguistics pp: 1-5.
20. Agirre E, De Lacalle OL, Fellbaum C, Marchetti A, Toral A, et al. (2009) Semeval-2010 task 17: All-words word sense disambiguation on a specific domain. In Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions, Association for Computational Linguistics pp: 123-128.
21. Alsewan Z (2015) Auflösung von sprachlichen mehrdeutigkeiten für spezifische dom en mittels simulated annealing. Bachelor thesis, Institute of Embedded Systems, Hamburg University of Technology, Hamburg, Germany.
22. Agirre E, Soroa A (2009) Personalizing pagerank for word sense disambiguation. In Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics pp: 33-41.

Citation: Abualhaija S, Zimmermann KH (2016) Solving Specific Domain Word Sense Disambiguation using the D-Bees Algorithm. Global J Technol Optim 7: 193. doi:10.4172/2229-8711.1000193