Parameter Transfer across Domains for Word Sense Disambiguation

Sallam Abualhaija
L3S Research Center
Leibniz University
abualhaija@l3s.de

Nina Tahmasebi
Department of Swedish
University of Gothenburg
nina.tahmasebi@svenska.gu.se

Diane Forin
Department of Applied
Mathematics & Modelling
Polytech Lyon
diane.forin@etu.univ-lyon1.fr

Karl-Heinz Zimmermann
Institute of Embedded Systems
Hamburg University of Technology
k.zimmermann@tu-harburg.de

Abstract

Word sense disambiguation is defined as finding the corresponding sense for a target word in a given context, which comprises a major step in text applications. Recently, it has been addressed as an optimization problem. The idea behind is to find a sequence of senses that corresponds to the words in a given context with a maximum semantic similarity. Metaheuristics like simulated annealing and D-Bees provide approximate good-enough solutions, but are usually influenced by the starting parameters. In this paper, we study the parameter tuning for both algorithms within the word sense disambiguation problem. The experiments are conducted on different datasets to cover different disambiguation scenarios. We show that D-Bees is robust and less sensitive towards the initial parameters compared to simulated annealing, hence, it is sufficient to tune the parameters once and reuse them for different datasets, domains or languages.

1 Introduction

Word sense disambiguation (WSD) is a well known problem in natural language processing (Agirre and Edmonds, 2007; Yarowsky, 1992). It is defined as the task of identifying the most likely meaning of a target word given the surrounding context. For example, in the sentence "The *mouse* of my computer is broken", the word *mouse* means "a computer device". WSD is an essential step in many applications like machine translation (Vickrey et al., 2005), lexical simplification (Specia et al., 2012) and many others (Ide and Véronis, 1998) and of particularly importance for creating accurate text understanding systems using the massive amounts of widely available text.

There are different methods to solve the WSD problem; supervised and knowledge-based methods rely on predefined sense inventories (Agirre and Edmonds, 2007; Martínez et al., 2007; Pedersen et al., 2005), and unsupervised methods that include sense induction and discrimination (Navigli, 2012; Brody et al., 2006; Schütze, 1998). Supervised methods generally achieve the best results (Navigli et al., 2007), however, they require strenuous effort to prepare annotated corpora or hand-crafted sense inventories. In addition, it is a process that has to be repeated for every language and domain especially since words change their senses over time (Tahmasebi et al., 2011; Kulkarni et al., 2015). In this paper, we rely on knowledge-based methods for WSD that make use of an existing knowledge-base like WordNet (Miller et al., 1990) and do not require annotated corpora. Our method can however work with any sense inventory, be a collaboratively created resource like Wiktionary [1] or an induced set of senses.

We will address WSD as an optimization problem (Schwab et al., 2012; Pedersen et al., 2005); Given a sequence of words as an input, a corresponding sequence of senses with the maximum semantic similarity should be returned as the output. The straight forward method considers all possible sense combinations in the context window to find the exact optimal solution. However, this method suffers from the combinatorial explosion problem, where the time complexity grows exponentially with the input size. So, WSD as an optimization problem is considered NP-hard (Dorigo and Stützle, 2004).

Approximate algorithms (metaheuristics) are

[1] https://www.wiktionary.org/

used to overcome the complexity problems and find near-optimal solutions while exploring as little of the search space as possible. Different metaheuristics have been applied to WSD, including but not limited to, the ant colony algorithm (ACA) (Schwab and Guillaume, 2011), simulated annealing (SA) (Cowie et al., 1992), genetic algorithm (GA) (Zhang et al., 2008) and D-Bees (Abualhaija and Zimmermann, 2016). The results have been good (Schwab et al., 2012), but the major concern is that they are sensitive towards their initial parameters due to their stochastic behavior. Therefore, to guarantee a good performance, parameters should be estimated for every language and domain beforehand.

In this paper, we study the parameter estimation of two metaheuristics for WSD, namely D-Bees (Abualhaija and Zimmermann, 2016) and SA (Cowie et al., 1992). We hypothesize that if metaheuristics are stable with respect to the initial parameters, the parameters can be tuned once on some dataset and then used on other datasets regardless of the domain or language. We test our hypothesis by studying how the best parameter settings for D-Bees and SA perform when tested on another domain.

Next, we explain the background. In Sec. 3, we glue the different pieces of this work together in one pipeline. Experiments and results are presented in 4. In Sec. 5 the results are further analyzed and compared. Finally, we conclude and give an outlook on future work in Sec. 6.

2 Background

2.1 WSD as an Optimization Problem

The WSD problem can be formulated as follows (Pedersen et al., 2005; Abualhaija and Zimmermann, 2016). Given a sequence of n words $W = (w_1, w_2, \ldots, w_n)$, a sequence of senses $\sigma = (s_1, s_2, \ldots, s_n)$, with a corresponding sense s_i for each word w_i, $1 \leq i \leq n$. Let $\mathcal{S} = \{\sigma_1, \ldots, \sigma_m\}$ be the set of all sequences of senses corresponding to all sense combinations of the words in the context window (the search space). The objective function is then $\mathrm{argmax}_{\sigma \in \mathcal{S}} \ell(\sigma)$, where ℓ is the score assigned to a sequence of senses. In our experiments, the score is calculated using a variant of the Lesk algorithm (eLesk) (Banerjee and Pedersen, 2002). This optimization approach has the advantage that all the words in the context window are disambiguated simultaneously.

2.2 Metaheuristics for WSD

We choose D-Bees and simulated annealing algorithms because the first has a comparable performance on both English and German (Abualhaija et al., 2017), while the latter is considered a baseline for many metaheuristics since it has a solid formulation, performed well on the SemEval 2007 dataset, and has few parameters to tune (Schwab et al., 2012; Tchechmedjiev et al., 2012).

The D-Bees Algorithm

D-Bees is inspired by the bee colony optimization approach (Teodorović, 2009). Given a sequence of n words, one word is chosen to represent a (virtual) hive, i.e., the word from which the search starts. Only the hive produces bee agents, and sends them to explore the search space, each holding one of its senses and looking for similar senses from the context words.

The D-Bees algorithm comprises of several forward and backward passes. In a forward pass, the bee agents construct a partial solution incrementally by appending a sense from the next word based on the sense frequency and then updating the quality according to the similarity value between the senses until a predefined number of moves is reached. After that, the bee agents return to the hive to exchange information initiating the backward pass.

In the hive, each bee agent performs two probabilistic decisions based on the qualities of their partial solutions. Loyalty decides whether the bee agent exploits its partial solution further or abandons it. In case of abandoning it, another partial solution should be followed (recruitment). The bee agents with good partial solutions act as recruiters and be followed. Following means that the two bee agents fly together in the next forward pass until the point where they stopped last time, from there on they can act independently.

These two passes are alternated until no more target words are to be disambiguated. The bee agent with the best found solution is stored. After a given number of iterations the best solution is returned as the output. More details can be found in (Abualhaija and Zimmermann, 2016).

Simulated Annealing

Simulated Annealing (SA) was first applied to the WSD problem by Cowie et al. (1992) and reimplemented by Schwab et al. (2012). Given a sequence of n words, SA starts with an initial sequence (σ)

by assigning a sense to each word in the sentence; either randomly or based on some heuristic; e.g., using the most frequent sense.

All the definitions of the senses and the related senses are retrieved, preprocessed and stored in a list of lemmas. For this list, the redundancy (R) is calculated as $\sum(f-1)$, such that f represents the frequency of a lemma occurs in the list. Each solution has an associated energy function (E) to be minimized, and is defined as $E = \frac{1}{1+R}$, where R is the redundancy.

At each iteration, the solution is modified by changing a sense of a randomly chosen word and creating a new solution, which is accepted by a probability $p = \exp^{-\Delta(E)/T}$, where $\Delta(E)$ is the difference between the energy values of the new and old solutions, and T is the temperature. The temperature is decreased by a cooling factor λ after each iteration. This process is repeated until a stopping criterion and the best found solution is returned.

2.3 Parameter Tuning

Parameter tuning is defined as a meta-optimization problem (Talbi, 2009). Given a base algorithm (e.g. D-Bees), and evaluation metric, the parameter estimation algorithm is used to find the parameter configuration of the base algorithm which performs the best for a specific dataset and domain.

In this work we use the Focused Iterated Local Search (ILS) (Hutter et al., 2007; Montero et al., 2014), which is a stochastic local search-based method suitable to tune stochastic metaheuristics. The intuition behind is to test all the possible parameter configurations by changing one parameter value at a time (one-exchange concept), and report the parameter configuration that results in the best performance.

Focused ILS starts with an initial parameter configuration and modifies it randomly. It uses a fixed number of random moves for perturbation, and always accepts the better or equally good parameter configurations. However, it reinitializes the search at random with some probability.

The parameter estimation is performed on a sample of the dataset (training set). The resulting best found parameter configuration is then evaluated on an independently sample of the dataset, the test set. The parameter estimation problem for WSD is defined formally by Tchechmedjiev et al. (2012).

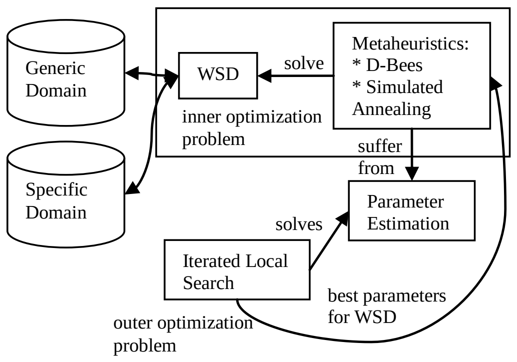

Figure 1: Parameter estimation meta-optimization diagram for WSD problem.

3 Optimizing the Optimizers

Fig. 1 shows the recursive view of the problem. That is, metaheuristics like D-Bees and SA are used to solve WSD as an optimization problem. However, to get the best performance of these algorithms, the initial parameters should be tuned beforehand. The senses in a generic domain dataset usually follow a different distribution than the ones in a specific domain where rare senses (or rather jargons) tend to be more prominent. Therefore, we designed the experiments to test parameters (i) within one dataset and (ii) across domains.

Parameters of the Metaheuristics

D-Bees has three main parameters that influence its performance. First, the hive-mode for choosing the hive and has two possible values: {"mini", "random"}. "mini" considers a word that has a minimal number of senses to limit the search and speed up the convergence whereas "random" refers to choosing uniformly at random. Second, the number of moves in a forward pass (c) : $c \in \{n/3, n/2\}$ where n is the number of words. This parameter controls the exploration process. So, using $n/2$ as an upper-bound ensures focusing on the good solutions early on and avoids prolonging the convergence. Finally, the number of recruiting bee agents (r) : $r \in \{b/3, b/2, 3 \cdot b/4, b-1\}$, b is the number of bee agents. The upper-bound $b-1$ reflects responding to better solutions quickly, as honeybees in nature.

SA has two parameters for which the values are chosen inspired by Cowie et al. (1992); Schwab et al. (2012). The temperature $T \in$ {70, 200, 400, 600, 700, 800, 900, 1000, 1100, 1200} and the cooling factor $\lambda \in \{0.1, \ldots, 0.9\}$.

4 Experiments and Results

The task is to estimate good parameters for each algorithm on the WSD problem by maximizing the F-measure. The best parameters are then tested on a test set and the results are compared to the most frequent sense baseline (MFS). In order to arrive to a valid conclusion, we use the Wilcoxon non-parametric statistical test.

4.1 Datasets

Tuning experiments are conducted using two datasets from the SemEval campaign. SemEval 2007 task 7 (Navigli et al., 2007) considers a coarse-grained disambiguation and is composed of five different texts (d001 – d005). Apart from d004, the corpus is general domain, and the precision of the MFS baseline is 78.89%. Second is SemEval 2010 task 17 (Agirre et al., 2009) which contains three texts (en1 – en3) on a specific domain (environment). The F-measure of the MFS baseline is 50.50%.

In our experiments we split each dataset into a training and a test set. For SemEval 2007 we used d001 or d004 as a training set and the rest for testing, while for SemEval 2010 we used en1 for training and the rest for testing.

4.2 Training Parameters (intra-Domain)

We start by comparing the parameters when trained and tested on the same corpus. The D-Bees results are compared against the MFS baseline because both introduce a bias on sense frequency when selecting the sense of the next word, also there is no previous literature on D-Bees. The baseline for SA is reported in (Schwab et al., 2012), this way we can test the parameter estimation algorithm compared to what was reported in the literature.

Fig. 2 shows the results of the top-10 parameters using d001, and tested on the rest of the texts of SemEval 2007. It can be proved that the top 10 parameters perform significantly better on the test set of the same corpus than the baseline with a confidence interval of 90% in both cases. Similarly, Fig. 3 and Fig. 4 summarizes the D-Bees and SA parameters behavior on SemEval 2007 dataset excluding d004 and SemEval 2010 excluding en1, respectively(d004 and en1 are used as training sets). For D-Bees, the baseline is the default parameter found on d001 in order to get an impression about how parameters trained on general domain perform on a specific one.

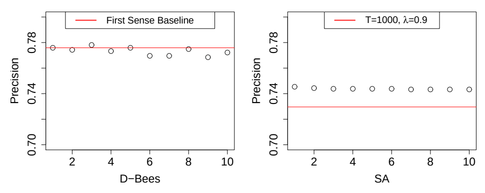

Figure 2: Top-10 parameters on SemEval 2007 excluding d001.

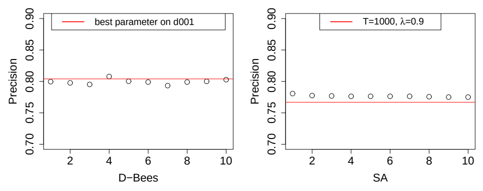

Figure 3: Top-10 parameters on SemEval 2007 excluding d004.

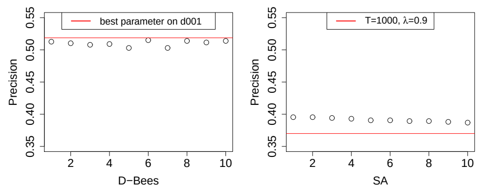

Figure 4: Top-10 parameters on SemEval 2010 excluding en1.

It can be observed that the behavior of the best parameters found on the two datasets do not vary for both algorithms.

4.3 Transferring Parameters (inter-Domain)

Next we test how parameters perform when transferred across domains and datasets. If the parameters that are trained on corpus (or domain) A perform better than a certain baseline for corpus (or domain) B, then these parameters are suitable for use. It follows that the algorithm is less sensitive towards the dataset, or the domain of dataset. Therefore, the parameters do not necessarily have to be re-estimated. Otherwise the baseline is sufficient.

Fig. 5 shows the top-10 parameters found by training on d001 of SemEval 2007 and testing on SemEval 2010 dataset. The baseline for D-Bees corresponds to the best parameter trained on en1

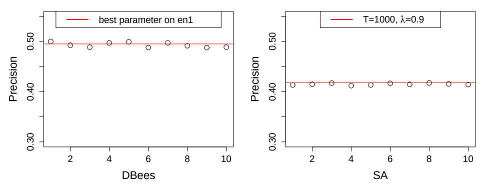

Figure 5: Top-10 parameters on d001 tested on SemEval 2010.

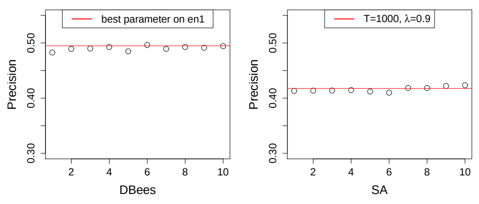

Figure 6: Top-10 parameters on d004 tested on SemEval 2010.

and tested on the rest of SemEval 2010 dataset. For SA, the baseline is the same as in the previous experiments which corresponds eventually to training the parameters on d001 from SemEval 2007 corpus. This makes the baselines for both algorithms equivalent.

It can be observed that some of the parameters of D-Bees significantly outperform the baseline, while for SA, the baseline performs best. Training the parameters of D-Bees on a generic domain then using them on another specific one is performing well. On the contrary, SA appears to be more sensitive towards the dataset.

In Fig. 6, the parameters are trained on d004 of SemEval 2007, considering d004 text as a specific domain on "computer science" and tested on the SemEval 2010. As expected, using the Wilcoxon test the performance of the parameters found on en1 are on par with those found on d004. It follows that training parameters on a specific domain dataset and then testing on a different specific domain one might work, but with a low certainty.

The last case is testing how well the parameters perform on a generic domain dataset (SemEval 2007) when trained on a specific domain from another dataset (SemEval 2010). Fig. 7 shows the results of this case, where the baseline outperforms all the parameters significantly for both algorithms. This supports our earlier argument that senses might have different distributions in specific domains.

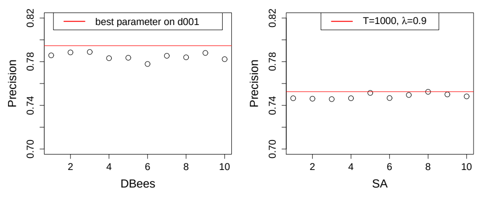

Figure 7: Testing the top-10 en1 parameters on SemEval 2007 dataset.

5 Analysis and Discussion

In this section, we discuss the results of solving the recursive problem, that is solving WSD using metaheuristics. In the first part we show the performance of using the best found parameters on the corresponding SemEval task. Then, we discuss the results of these parameters with the insight of their performance on the dedicated task.

5.1 Influence on WSD Results

Going back to the original problem (WSD), we compare the performance of D-Bees with the other participating systems reported in both SemEval 2007 and 2010 tasks. We focus on D-Bees and not SA because the former performs better in both parts of the recursive problem, WSD as well as parameter estimation. Also we apply D-Bees on the whole dataset to obtain a fair comparison with the other reported systems in the task.

Tab. 1 shows the performance of D-Bees on SemEval 2007 (Navigli et al., 2007). It can be observed that the results of D-Bees are outperformed mainly by supervised systems and are comparable to the first sense baseline (MFS). In general texts MFS disambiguates the target words to a large extent. Therefore, the chance to hit the correct meaning is high. Unlike MFS, D-Bees considers other frequent senses as well, which might match more precise meanings of the words provided that the definitions of the senses are close to each other because WordNet is fine-grained. This makes the D-Bees algorithm promising.

Tab. 2 shows the performance of D-Bees on SemEval 2010 using both best found parameters on the generic and specific domain. We report the precision and recall similar to the task (Agirre et al., 2009). The two D-Bees versions perform somewhat close although none beats the MFS baseline. The D-Bees (specific) is the next best system. This supports the conclusion that parameters trained on a general domain can be applied

Table 1: Comparison of D-Bees with 100% coverage evaluated on Semeval 2007 Task 7 († ≡ Supervised Method).

System	F-Measure (%)
UoR-SSI†	83.21
Nus-PT†	82.5
Nus-ML†	81.58
LCC-WSD†	81.45
GPLSI†	79.55
D-Bees	79.46
MFS Baseline	78.89
UPV-WSD†	78.63
Degree (Babelnet)	77.01
PageRank (Babelnet)	72.60
TKB-UO	70.21
RACAI-SYNWSD	65.71

Table 2: A comparison of D-Bees with the knowledge-based systems on SemEval 2010 Task 17.

System	Precision (%)	Recall (%)
MFS Baseline	50.5	50.5
D-Bees (specific)	50.0	50.0
CFILT-3	51.2	49.5
Treematch	50.6	49.3
D-Bees (generic)	48.9	48.9
kyoto-2	48.1	48.1
RACAI-MFS	46.1	46.0
UCF-WS	44.7	44.1
HIT-CIR-DMFS-1.ans	43.6	43.5
IIITH2-d.r.l.baseline.05	49.6	43.3

on a specific one with a decent performance. The other way around however, does not work.

5.2 Parameter Tuning

Our experiments show that D-Bees achieves F-score of 79.46% and 48.9% on SemEval 2007 and 2010 respectively using parameters trained on a generic domain. Adjusting the parameter to the specific domain increases the performance to 50.0% on SemEval 2010, the best of all evaluated systems. While SA achieves 24.0% and 40.0% using the best parameters trained on SemEval 2007 and SemEval 2010 respectively. We find that D-Bees performs comparable to the state-of-the-art (Abualhaija and Zimmermann, 2016; Schwab et al., 2012). The results of SA indicate that it is sensitive towards its parameters, thus have to be tuned beforehand for every domain and language.

Lemmas and part of speech tags of the words are not offered in SemEval 2010, so the performance of the WSD system depends partially on the lemmatizer and POS-tagger. Also, errors in pre-processing lead to an unstable behavior of D-Bees. Nevertheless, D-Bees shows a robust behavior by achieving scores on par with the MFS and the best systems. For the specific domain, the parameters of D-Bees perform better when the number of recruiters is greater than at least half of the bee agents, unlike on a generic domain where a third is sufficient.

The sensitivity of both algorithms is further analyzed on the lexical substitution task for English and German (Abualhaija et al., 2017). For D-Bees we used the parameters found on SemEval 2007 dataset (d001) and the default parameters as given by Cowie et al. (1992) for SA. D-Bees achieves competitive results for both languages, while the performance of SA could be improved by tuning the parameters on the testing language. This supports the conclusion that D-Bees algorithm is less sensitive towards its initial parameters.

6 Conclusion and Outlook

In this paper, we tested the hypothesis that metaheuristics are good approximation algorithms to solve the WSD optimization problem. We compared D-Bees to SA and found that D-Bees performs better for WSD. We studied the parameter tuning problem since metaheuristics could be sensitive towards their initial parameters. We found that transferring the parameters across domains (i.e., training on one domain and testing on another) works well for D-Bees, which reduces the need to tune the parameters on each new WSD scenario and dataset.

In some sense, we could find robustness in the parameters across datasets (D-Bees best performance compared to SA) but more limited transferability across domains. We believe that this difficulty can be avoided by including knowledge about the corpus domain from the lexical resource. D-Bees beats most unsupervised methods in both SemEval 2007 and 2010 contests. This makes it suitable for solving the WSD problem. Our experiments deduced clues on how to choose parameters for specific domain.

Acknowledgement

This work was partially funded by the German Federal Ministry of Education and Research (BMBF) under the project K3 (13N13548).

References

Sallam Abualhaija, Tristan Miller, Judith Eckle-Kohler, Iryna Gurevych, and Karl-Heinz Zimmermann. 2017. Metaheuristic approaches to lexical substitution and simplification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*. Association for Computational Linguistics.

Sallam Abualhaija and Karl-Heinz Zimmermann. 2016. D-bees: A novel method inspired by bee colony optimization for solving word sense disambiguation. *Swarm and Evolutionary Computation* 27:188 – 195.

Eneko Agirre, Oier López De Lacalle, Christiane Fellbaum, Andrea Marchetti, Antonio Toral, and Piek Vossen. 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, pages 123–128.

Eneko Agirre and Philip Glenny Edmonds. 2007. *Word sense disambiguation: Algorithms and applications*, volume 33. Springer Science & Business Media.

Satanjeev Banerjee and Ted Pedersen. 2002. An adapted lesk algorithm for word sense disambiguation using wordnet. In *Computational linguistics and intelligent text processing*, Springer, pages 136–145.

Samuel Brody, Roberto Navigli, and Mirella Lapata. 2006. Ensemble methods for unsupervised wsd. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 97–104.

Jim Cowie, Joe Guthrie, and Louise Guthrie. 1992. Lexical disambiguation using simulated annealing. In *Proceedings of the 14th conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, pages 359–365.

Marco Dorigo and Thomas Stützle. 2004. *Ant Colony Optimization*. Bradford Company, Scituate, MA, USA.

Frank Hutter, Holger H Hoos, and Thomas Stützle. 2007. Automatic algorithm configuration based on local search. In *AAAI*. volume 7, pages 1152–1157.

Nancy Ide and Jean Véronis. 1998. Introduction to the special issue on word sense disambiguation: the state of the art. *Computational linguistics* 24:2–40.

Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2015. Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, pages 625–635.

David Martínez et al. 2007. Supervised corpus-based methods for wsd. In *Word Sense Disambiguation*, Springer, pages 167–216.

George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to wordnet: An on-line lexical database*. *International journal of lexicography* 3:235–244.

Elizabeth Montero, María-Cristina Riff, and Bertrand Neveu. 2014. A beginner's guide to tuning methods. *Applied Soft Computing* 17:39–51.

Roberto Navigli. 2012. A quick tour of word sense disambiguation, induction and related approaches. In *International Conference on Current Trends in Theory and Practice of Computer Science*. Springer, pages 115–129.

Roberto Navigli, Kenneth C Litkowski, and Orin Hargraves. 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, pages 30–35.

Ted Pedersen, Satanjeev Banerjee, and Siddharth Patwardhan. 2005. Maximizing semantic relatedness to perform word sense disambiguation. *University of Minnesota supercomputing institute research report UMSI* 25:2005.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational linguistics* 24:97–123.

Didier Schwab, Jérôme Goulian, Andon Tchechmedjiev, and Hervé Blanchon. 2012. Ant colony algorithm for the unsupervised word sense disambiguation of texts: Comparison and evaluation. In *COLING*. pages 2389–2404.

Didier Schwab and Nathan Guillaume. 2011. A global ant colony algorithm for word sense disambiguation based on semantic relatedness. In *Highlights in Practical Applications of Agents and Multiagent Systems*, Springer, pages 257–264.

Lucia Specia, Sujay Kumar Jauhar, and Rada Mihalcea. 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, pages 347–355.

Nina Tahmasebi, Thomas Risse, and Stefan Dietze. 2011. Towards automatic language evolution tracking, a study on word sense tracking. In *Joint Workshop on Knowledge Evolution and Ontology Dynamics*.

El-Ghazali Talbi. 2009. *Metaheuristics: From Design to Implementation*. Wiley Publishing.

Andon Tchechmedjiev, Jérôme Goulian, Didier Schwab, and Gilles Sérasset. 2012. Parameter estimation under uncertainty with simulated annealing applied to an ant colony based probabilistic wsd algorithm. In *Proceedings of the First International Workshop on Optimization Techniques for Human Language Technology*. pages 109–124.

Dušan Teodorović. 2009. Bee colony optimization (bco). In *Innovations in swarm intelligence*, Springer, pages 39–60.

David Vickrey, Luke Biewald, Marc Teyssier, and Daphne Koller. 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, pages 771–778.

David Yarowsky. 1992. Word-sense disambiguation using statistical models of roget's categories trained on large corpora. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*. Association for Computational Linguistics, Stroudsburg, PA, USA, COLING '92, pages 454–460.

Chunhui Zhang, Yiming Zhou, and Trevor Martin. 2008. Genetic word sense disambiguation algorithm. In *Intelligent Information Technology Application, 2008. IITA'08. Second International Symposium on*. IEEE, volume 1, pages 123–127.