UNIVERSITATEA DIN PITEŞTI

BULETIN ŞTIINŢIFIC

S e r i a

Matematică şi Informatică

EXTRAS

18

EDITURA UNIVERSITĂŢII DIN PITEŞTI
2012

Buletin Ştiinţific - Universitatea din Piteşti
Seria Matematică şi Informatică, Nr. **18** (2012), pg.45-54

An Overview for Speeding Up a Knowledge-based Word Sense Disambiguation Method

Andrei MINCĂ*

December 2012

Abstract

This work investigates means of speeding up the computation of a knowledge-based Word Sense Disambiguation (WSD) method derived from the Lesk algorithm. This method, designed in the SenDiS (Sense Disambiguation System) research project [4], has several computational steps which are suitable for parallelization, distributed or heterogeneous computation. Depending of the step concerned, the methods considered spread from the simple SIMD (single instruction, multiple data) computation model to the use of heterogeneous architectures. We use WordNet tagged glosses [3] to build the lexicon network required in our WSD approach, and the corpuses Senseval2, Senseval3 and SemCor corpuses [1], [2] are used for testing.

2010 Mathematics Subject Classification: Primary 68T50, 65Y05; Secondary 68Q85.

Key words and phrases: knowledge-based word sense disambiguation, Lesk algorithm, WordNet tagged glosses, GPU computing, distributed computing, heterogeneous computing, cloud on a chip

*R&D Department, SOFTWIN, Romania; aminca@softwin.ro

1 Introduction

Word sense disambiguation (WSD) is the process of establishing the concept that a given use of a word represents. Usually, WSD is computed in natural language processing systems by examining semantic domains for each word with various restriction mechanisms.

A process that deals with natural entities, such as languages, is always bound to have a high complexity. Often, this complexity grows at least exponentially in the size of the problem, both is space and computation time. Trying to reduce the complexity by approximate algorithms usually leads to inaccurate solutions. Since we aim for precise solutions, we investigate ways of using the power of parallel or distributed computation in order to decrease the time or space used by these demanding WSD algorithms.

Even if it is considered to be AI-complete, the ongoing advances in parallel computing architectures and NLP (Natural Language Processing) algorithms keep on improving the solutions accuracy for the WSD problem.

The transition from single-core to multi-core CPUs in mainstream machines, requires changes throughout the software stack from operating systems to languages, to tools, and permanently affects the way software developers have to write code in order for applications to continue exploiting Moore's dividend. Just recently the transition to parallel computing was achieved in all mainstream form factors, with the arrival of multi-core tablets and smartphones [5].

Taking into consideration the above, our work focuses on means to speed up the computational processes involved in a specific WSD method, by employing the advantages of current and near-future architectures.

The paper is organised as follows. In Section 2, we present related research and work that make the basis of this research report. In Section 3, we further describe the problem and the challenges. In Section 4, we identify the granularity and the parallel computation suitability for each step in the WSD method. Section 5 concludes the current research activity.

2 Previous work

The WSD method is based and derived from an assumption made in [6]. There, we find the most general and well-known attempt to utilize information in machine-readable dictionaries for WSD, that of Lesk (1986). The Lesk algorithm computes a degree of overlap, i.e., number of shared words, in definition texts of words that appear in a ten-word window of context. The sense of a word with the greatest number of overlaps with senses of other words in the window is chosen as the correct one.

Many variations of the Lesk algorithm have been proposed, including: i) versions that attempts to solve the combinatorial explosion of possible word sense combinations when more than two words are considered; ii) algorithm variations where each word in a given context is disambiguated individually, by measuring the overlap between its corresponding dictionary definitions and the current sentential context; and iii) alternatives where the semantic space of a word meaning is augmented with definitions of semantically related words. All these different versions reported encouraging results.

For example, in [7] the Lesk algorithm is adapted to the online lexical database called WordNet [8], freely distributed by Princeton University. Besides storing words and their meanings, WordNet also defines a rich set of relationships between words. For instance, it says that a dog is a canine which is a carnivore. Or, it tells that to bark is one way to interact, and that one sense of bark is a kind of a covering. It says that pine and cone are two kinds of coniferous trees while sandwiches form a part of breakfast. The use of these relations significantly improves the disambiguation process and also increases the base problem complexity.

Further on, [9] describes a complex language for abstracting grammar, named GRAALAN, which can store even richer sets of word or word sense relations. More precisely, besides the usual relations already present in WordNet-like projects, GRAALAN deploys also relation types like *extension from, extension in, reduction from, reduction in, figurative of, literal for, abstract for* and others.

All the above were mentioned to illustrate the basis of the WSD method considered and the buildup of complexity when using knowledge based methods.

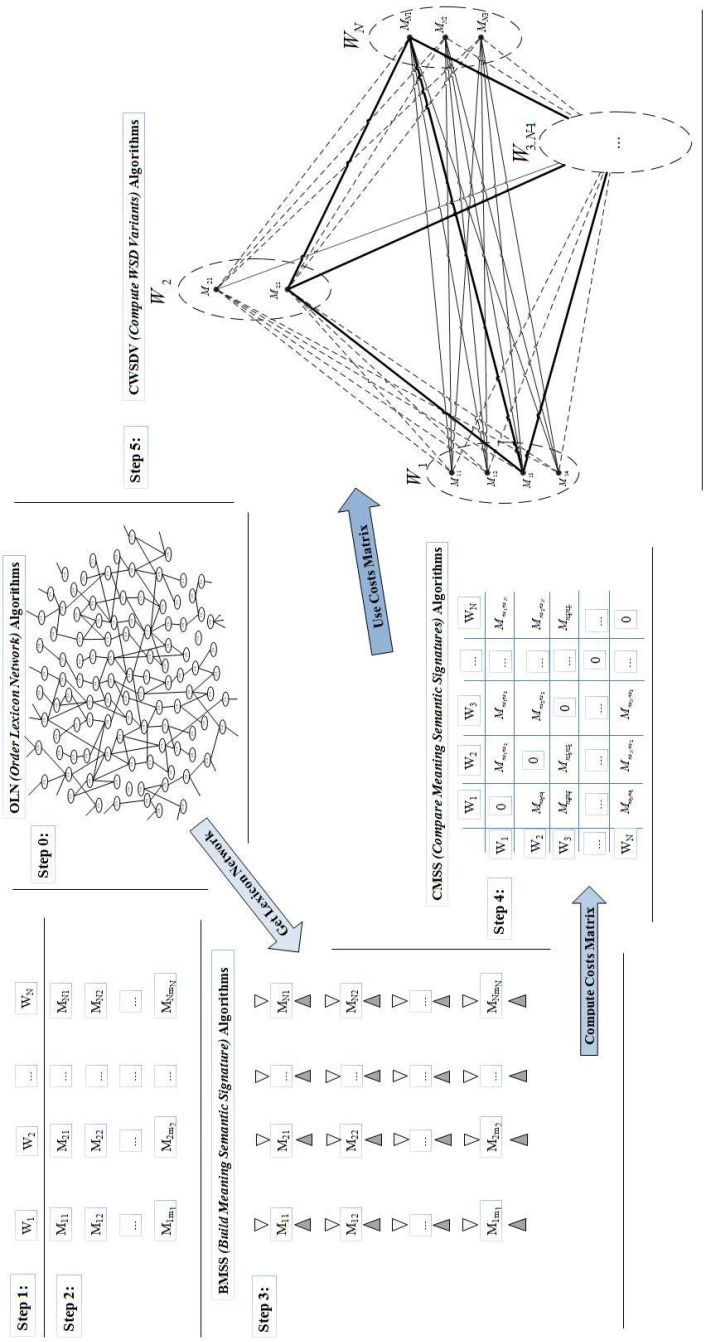


Figure 1: The main steps in our WSD method (see [10])

Our previous work [10], which is the WSD model and precursor for the current WSD method, acknowledges all the existing rich information and proposes only the use of semantically tagged glosses type of relations and a lexicon network built on such relations.

Given a target text, as mentioned in [10] and shown in Fig. 1, for the SenDiS system we identify well defined and explicit steps or stages:

- pre step for transformation of the lexicon network
- assignation of target words
- identification of word meanings against the lexicon
- building of semantic signature for every meaning
- computation of similarity costs between semantic signatures of word meanings
- final step of computing word sense disambiguation considering:
  - similarity costs for semantic signatures pairs
  - word context window
  - text context window
  - target words topologic order and/or various filters

Also we can identify four well-defined algorithm types:

- **OLN** (Order Lexicon Network) algorithms - various transformations that can be performed on the lexicon network in order to optimize it, the two major types: truncation and leveling.
- **BMSS** (Build Meaning Semantic Signature) algorithms - building of semantic signatures (such as trees, sets) for word senses using lexicon or semantic networks.
- **CMSS** (Compare Meaning Semantic Signatures) algorithms - this type of algorithm computes the similarity between two given meanings by matching their semantic signatures.
- **CWSDV** (Compute WSD Variants) algorithms - the use of computed similarity costs for word meanings to achieve the final step of WSD.

3 Challenges

As presented above, the SenDiS system uses the collaboration of four distinctive algorithm types to achieve WSD. Doing so, the system is modularized and flexible for trying various combinations and algorithm behaviors.

The system calibration, for any language, is accomplished by computing all the possible combinations and extracting the best results. A combination is a tuple of the form (a lexicon network, OLN algorithm, BMSS algorithm, CMSS algorithm, CWSDV algorithm, test corpus).

Below, we summarise characteristics of the tuple components that emphasize the complexity of a system calibration.

Lexicon network: A graph or directed graph of relations between word senses having from 100,000 to 250,000 vertices or word senses, and 1,500,000 to 3,000,000 edges or word senses relations.

OLN Algorithms: These are various types of a lexicon network transformation pursuing one or more of the following:

- lesser or no strong connected components;
- leveling the network so that we have:
  - a minimum or a maximum number of levels;
  - a minimum or a maximum number of primitives (a primitive is a basic sense/concept) or universals (an universal is a derivate sense/concept that is not used to form another concept);
  - a minimum number of edges eliminated.

There are three current implementations, but many more are ongoing.

BMSS Algorithms: These are various methods of building explanatory sets or semantic trees on the transformed lexicon network. An explanatory set or a semantic tree is a semantic signature of a sense/meaning against a lexicon network.

There are two current implementations, but many more are on-going.

CMSS Algorithms: These consist of various ways of computing semantic similarities between two meaning semantic signatures. There can be $(N-1) \cdot N/2$ matching combinations assuming that all N meanings are desired to be matched to all other meanings.

There are two current implementations, but many more are on-going.

CWSDV Algorithms: Due to the availability of semantic similarities costs there can be multiple ways that can be imagined for finally achieving the WSD. Of course, these similarities can be combined with different semantic information of various types. There can be $N_1 \cdot N_2 \cdot ... \cdot N_w$ combinations for WSD solutions, where w is the number of target words.

There is one current algorithm implementation.

Test corpus: A standard international acknowledged corpus, that is annotated with word senses beside other semantic information, which can be used to test WSD methods.

The SemCor corpus is composed of 37,176 sentences, grouped in 352 contexts, containing 779,294 tokens out of which 234,843 tokens are sense-tagged.

All the above present the complexity and the computing resources needed for a system calibration. The running time of one single configuration on a common system can last for several hours or more depending of the characteristics of each tuple component. For example, sequential computation time for testing SemCor corpus is currently at 120 hours with our best algorithms.

There is also interest in speeding up the WSD process for the real-time type applications. When dealing with short phrases the time is reasonable enough, but for larger paragraphs or when trying to achieve WSD for large texts the computation grows at least exponentially. The costs involved in such improvements, whatever form they may take, are not to be ignored.

4 Speed-up potential

For each SenDiS computation step we identified a degree of speed-up potential that can be operated using different parallel, accelerated or distributed computing paradigms. Below each table, we present the chosen parallel programming model and possibly an existing dedicated architecture or a mixture that can be exploited with better results.

OLN stage:

Speed-up potential	Granularity	Data dependency
low	low	high

This stage fits the least the parallelization due to the strong data dependencies inside a dense graph as a lexicon network can be. Though, for this step, the Heartbeat model or the separate analysis of Strong Connected Components can be applied for parallelization.

BMSS stage:

Speed-up potential	Granularity	Data dependency
high	medium	low

Building semantic signatures for meanings against a lexicon network can be done locally with many core architectures or in a distributed

fashion inside a cluster or grid.

One could argue that in this stage, a BMSS algorithm type can be executed only once for a meaning. Indeed that can be the case. But the storage requirements will be large due to the many parameterisations of each algorithm and the size of such a semantic signature. This aspect will not be ignored once the system is considered calibrated.

As programming models we identified:

- OpenMP, which best fits on multi-core/many-core processors or for cluster computing;
- MPI, best for distributing computation on cluster or grid architectures;
- TBB, best for a multi-task approach on multi-core/many-core processors.

CMSS stage:

Speed-up potential	Granularity	Data dependency
high	medium-high	low

This stage is very suitable for speed up using both the SIMD (single instruction, multiple data) and the MIMD (multiple instruction, multiple data) approaches. What is interesting is that both levels of speed-up can be operated on the new and promising dense core architectures due to the medium to high granularity of the problem and to low data dependency.

The considered dense core architectures are the next generations of NVIDIA GPUs, the IBM Cell processor and the Intel Xeon Phi. Each approach provides a model of lower speed cores, but greater in number. Also, the number of cores is inversely coupled with the worker speed. These models are good for problems with high granularity and low data dependencies.

The solution from Intel, Xeon Phi, has the advantage that is x86 compatible and there is no need for language directives to map the software on the architecture. Moreover, Xeon Phi is announced as "cloud on a chip" architecture being a general purpose co-processor with high FLOPs capabilities (over 1 TeraFLOPs with 50+ cores).

CWSDV stage:

Speed-up potential	Granularity	Data dependency
medium-high	high	medium

Computing the best WSD variant(s) can be really computationally demanding. For a reasonable text size and an All-Words WSD method there can be millions or even a greater number of combinations. Of

course, one could apply intelligent filters or a specific topologic order for words but the complexity decreases only slightly.

Due to the high granularity and a medium to low data dependency, this stage fits very well a speed-up approach. The programming models that are promising are OpenMP on any multi-core CPU architecture, CUDA on NVIDIA GPUs and TBB on Intel Xeon Phi.

5 Conclusions

Based on the very detailed investigations above, we conclude that the SenDiS WSD method is suitable for parallelization (see the tables in the previous section) and this can lead to a significant speed-up in the computational process.

The following table summarizes the speed-up potential indicators of the system described in the previous section. We identified, for each step, the indicators low (L), medium (M), or high (H) when considering a SIMD (aka S/D) approach and/or a MIMD (aka M/D) approach, with respect to speed-up potential, granularity, and data dependency. Moreover, we matched the most suitable speed-up approaches in the following dimensions: API model, hardware model (HW model) , development cost (Dev cost), and hardware cost (HW cost).

Speed-up expectation								API model	HW model	Dev cost	HW cost
OLN		BMSS		CMSS		CWSDV					
S/D	M/D	S/D	M/D	S/D	M/D	S/D	M/D				
M	L-M	M-H	-	M	L	M	L	OpenMP	Cluster	L-M	M-H
-	-	M-H	-	L	-	M	L	MPI	Cluster	M	M-H
-	-	M-H	-	L	-	L-M	-	MPI	Grid	M-H	H
-	-	L	-	M-H	M-H	-	-	CUDA	NVIDIA GPU	M-H	L
-	M	M	L	M	L-M	M	M-H	CBE SDK	IBM CELL	M-H	L
M	M-H	H	L-M	H	M-H	M	H	OpenMP, TBB	Xeon Phi	L	L

Table 1: Speed-up potential indicators

Acknowledgments This work was supported by the project "General Word Sense Disambiguation System applied to Romanian and English Languages" (SenDiS), co-funded by a Romanian research grant no. 207/20.07.2010 from the "Research, Technological Development and Innovation for Competitiveness" program.

References

[1] SensEval: `http://www.senseval.org`

[2] SemCor: `http://www.cse.unt.edu/~rada/downloads.html#semcor`

[3] Semantically Tagged glosses for WordNet:
`http://wordnet.princeton.edu/glosstag.shtml`

[4] Sense Disambiguation System:
`http://rd.softwin.ro/index.php/ro/proiecte/sendis`

[5] Welcome to the jungle:
`http://herbsutter.com/welcome-to-the-jungle`

[6] M. Lesk, *Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone*, Proceedings of SIGDOC 86, 1986

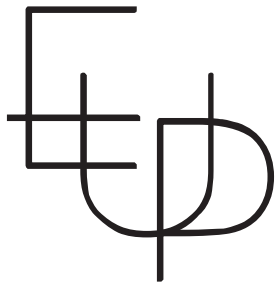
[7] S. Banerjee, T. Pedersen, *An adapted Lesk algorithm for word sense disambiguation using WordNet*, Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics, Mexico City, 2002

[8] WordNet at Princeton University: `http://wordnet.princeton.edu`

[9] St. Diaconescu, I. Dumitrascu, *Complex Natural Language Processing System Architecture, Advances in Spoken Language Technology*, The Publishing House of the Romanian Academy, Bucharest, 2007, pp. 228-240

[10] A. Minca, St. Diaconescu, *An approach to knowledge-based Word Sense Disambiguation using semantic trees built on a WordNet lexicon network*, The 6th Conference on Speech Technology and Human-Computer Dialogue (SpeD 2011), Brasov, Romania, 2011

[11] R. Navigli, *Word sense disambiguation, A survey.* ACM Comput. Surv. 41, 2, Article 10, February 2009



ISSN 1453 - 116x