IMPROVEMENT WSD DICTIONARY USING ANNOTATED CORPUS AND TESTING IT WITH SIMPLIFIED LESK ALGORITHM

Ahmed H. Aliwy[1] and Ayad R. Abbas[2]

[1]Department of Computer Science, University Of Technology, Baghdad, Iraq
ahmed_7425@yahoo.com
[2] Department of Computer Science, University Of Technology, Baghdad, Iraq
ayad_cs@yahoo.com

ABSTRACT

***WSD** is a task with a long history in computational linguistics. It is open problem in NLP. This research focuses on increasing the accuracy of Lesk algorithm with assistant of annotated corpus using Narodowy Korpus Jezyka Polskiego (NKJP "Polish National Corpus"). The NKJP_WSI (NKJP Word Sense Inventory) is used as senses inventory. A Lesk algorithm is firstly implemented on the whole corpus (training and test) and then getting the results. This is done with assistance of special dictionary that contains all possible senses for each ambiguous word. In this implementation, the similarity equation is applied to information retrieval using tf-idf with small modification in order to achieve the requirements. Experimental results show that the accuracy of 82.016% and 84.063% without and with deleting stop words respectively. Moreover, this paper practically solves the challenge of an execution time. Therefore, we proposed special structure for building another dictionary from the corpus in order to reduce time complicity of the training process. The new dictionary contains all the possible words (only these which help us in solving WSD) with their tf-idf from the existing dictionary with assistant of annotated corpus. Furthermore, eexperimental results show that the two tests are identical. The execution time - of the second test dropped down to 20 times compared to first test with same accuracy*

KEYWORDS

Corpus-based WSD, Lesk algorithm, dictionary and corpus based WSD.

1. INTRODUCTION

A **word sense disambiguation** is the task of determining the suitable sense for the ambiguous words in the context. It has a long history in computational linguistics, and is a non-trivial task result from the nature of language semantics. All the current used algorithms did not achieved high levels of accuracy. Many of these algorithms depend on contextual similarity for selecting the proper sense [1].

The revolution of the work on WSD may be start in 1980's where the digital large-scale lexical resources became widely available [2].

Tasks of many NLP improved using WSD because word sense ambiguities results in many problems in many applications such as, **information retrieval, machine translation, question**

answering, **information retrieval**, and **text classification** therefore; WSD is an important task. WSD algorithm is varying from one application to another [1]. In this paper, we ignored the applications and focuses on the implementation and evaluation of WSD systems as a stand-alone task.

Most of the used WSD algorithms are dictionary-based or corpus-based methods. Dictionary-based methods choose the sense whose gloss or definition shares the most words with the target word's neighborhood [1]. Lesk algorithm is an example of dictionary based methods. It is the most well-studied algorithm for sense disambiguation.

Corpus-based methods require an annotated data where each ambiguous word has the correct sense in each sentence. This annotation is done by intervention of human. Most of these methods are called "supervised" because they learn from previously sense annotated data [3]. Lesk Algorithm gives better results if it used with annotated corpus than without.

2. RELATED WORK

Our approach, as we will see, is improving the WSD dictionary using annotated corpus. The tests are made by using Lesk algorithm. There are many researches in WSD field. Most of them used dictionary based, corpus based and hybrid methods.

Montoyo et al. [4] presented two WSD methods based on two main methodological approaches: a knowledge-based method and a corpus-based method. Their approach combines various sources of knowledge, through combinations of the two WSD approaches as mentioned above. They showed how to combine these methods and sources of information in order to achieve good results in the disambiguation. Little combinations are based while some of them are voting.

Ledo-Mezquita et al. [5] proposed a method of word sense disambiguation based on the combination of the original Lesk method and the simplified one with additional application of large lexical resources, like synonym dictionaries, ontologies, etc. their experimental results show that the method has better precision than the baseline Lesk-based methods. In other side, Basile et al. [6] adopted the simplified Lesk algorithm to disambiguate adjectives and adverbs, combining it with other two methods for nouns and verbs.

Ponzetto and Navigli [7] try to maximize the chances of overlap between glosses or between the gloss and the context in Lesk algorithm by extended WordNet with Wikipedia pages.

Viveros-Jiménez et al. [8] proposed simple strategies for the context window selection that improve the performance of the Simple Lesk algorithm by: (1) constructing the window only with words that have an overlap with some senses of the target word, (2) excluding the target word itself from matching, and (3) avoiding repetitions in the context window.

Schwab et al. [9] proposed GETALP: an unsupervised WSD algorithm inspired by Lesk. A local similarity are computed using the classical Lesk measure (overlap between glosses), and then these local similarity is propagated to the whole text (global similarity) using an algorithm inspired by the Ant Colony. This approach was applied to English and Italian.

Basile et al. [10] described an algorithm which extends two well-known variations of the Lesk WSD method. The main contribution of them approach relies on the use of a word similarity function defined on a distributional semantic space to compute the gloss-context overlap. They used private dataset (BabelNet API 1.1.1) provided by the authors. They used two languages, Italian and English, in the tests.

3. Dictionaries Based And Thesauri WSD Methods

In 1980's, after the theses of Amsler's (1980) and Michiel's (1982), Machine-readable dictionaries (MRDs) became an important source of knowledge. The first attempts were to extract lexical and semantic knowledge bases from MRDs. Then, it became the basis of lexical semantic studies [2].

WordNet is an example of improved MRD. It encodes a rich semantic network of concepts therefore; it can be an important source of word senses in English language [1]]. MRDs rapidly became a staple of WSD research but they lack pragmatic information that enters into sense determination [2].

Thesauri provide information about relationships between words, like synonymy, antonymy and, possibly, further relations. Roget's International is the most widely used thesaurus in the field of WSD [11]. The most well-known algorithm deals with MRD for WSD is Lesk algorithm.

3.1. Lesk Algorithm

The Lesk algorithm uses dictionary definitions (gloss/examples) to disambiguate a polysemous word in a sentence context. The original Lesk algorithm measures overlap between sense definitions for all words in the text and identify simultaneously the correct senses for all words in the text. A word is assigned to the sense whose gloss shares the largest number of words in common with the glosses of the other words [12]. There are many varieties of Lesk algorithm, the famous are Original and simplified.

We can see the complexity of original Lesk algorithm from the following example: if we have nine open class words with following number of senses: 26, 11, 4, 8, 5, 4, 10, 8, 3 then the sense combinations is 43,929,600. It is huge number and hence practically difficult to implement. It is the main reason for using, other version of Lesk algorithm, simplified Lesk.

3.2. Simplified Lesk Algorithm

The simplified Lesk algorithm is a modified version of Lesk algorithm as shown in figure 1 [13]. It tries to solve each ambiguous word alone without regards for other senses for other words (words which have sense ambiguity in the same context). For this reason, it can be used in practice.

```
function Simplified_Lesk(word, sentence)
  best-sense←most frequent sense for word
  max-overlap←0
   context←set of words in sentence
   for each sense in senses of word do
      signature←set of words in the gloss_examples of sense
        overlap←Compute_overlap(signature, context)
        if overlap > max-overlap then
                  max-overlap←overlap
                  best-sense←sense
   end
   return(best-sense){ returns best sense of word}
```

Figure 1. simplified Lesk algorithm [1].

Simplified Lesk algorithm can be used with annotated corpus by:

- Additionally Use sentences in corpus to compute signature of senses.
- Computing weighted overlap by using *idf*.

4. SIMILARITY AND COMPUTING TF-IDF FROM TRAINING CORPUS

If there are *n* different words (types; e.g., the most frequent meaningful words in a language) then, a document *j* may be represented as:

$$\vec{d}_j = (tf_{1,j}, tf_{2,j}, tf_{3,j}, \ldots, tf_{n,j}) \quad (1)$$

Where $tf_{i,\,j}$ is the *term frequency* of the word *wi* in document *j*. Similarly, a query *q* may be represented as:

$$\vec{q} = (tf_{1,q}, tf_{2,q}, tf_{3,q}, \ldots, tf_{n,q}) \quad (2)$$

The simplest approaches are based on the bag-of-words (multiset of words) model. In such a model the similarity between the query and a document is the cosine of appropriate vectors:

$$sim(\vec{q}, \vec{d}_j) = \frac{\vec{q} . \vec{d}_j}{|\vec{q}| \times |\vec{d}_j|} = \frac{\sum_{i=1}^{n} tf_{i,q} \times tf_{i,j}}{\sqrt{\sum_{i=1}^{n} tf_{i,q}^2} \times \sqrt{\sum_{i=1}^{n} tf_{i,j}^2}} \quad (3)$$

The equation (3) can be used with Lesk Algorithm without annotated corpus.

Because we deal with annotated corpus we need to other factor which is Inverse document frequency (*idf*). Inverse document frequency *idf* assigns more weight to words which are more specific for the given document. For a word *wi*:

$$idf_i = \log \frac{N}{n_i} \quad (4)$$

Where *N* is the number of documents in the collection and *ni* is the number of documents containing the word *wi*.

A frequently used measure of the weight of word *wi* in document *j* is *tf.idf (tf-idf*, TFIDF)[1]. Then, the similarity can be computed as [1]:

$$sim(\vec{q}, \vec{d}_j) = \frac{\sum_{w \in q,d} tf_{w,q} tf_{w,j} (idf_w)^2}{\sqrt{\sum_{i=1}^{n} (tf_{i,q} idf_i)^2} \times \sqrt{\sum_{i=1}^{m} (tf_{i,j} idf_i)^2}} \quad (5)$$

Where

[1] There are numerous variants of *tf.idf*.

- $tf_{w,q}$ is term frequency of the word w in query q but $tf_{i,q}$ is term frequency of word wi in the query.
- $tf_{w,j}$ term frequency of the word w in document dj.
- idf_w is inverse document frequency of the word w in document j.
- *n* and *m* are number of the different words in query and document j respectively.

The above equation can be used to calculate the best word sense as following: Suppose a dictionary contains orthogonal entries[2]. Each entry has more than one senses and each sense has some gloss examples. The simple way, for selecting the best sense, is by checking the similarity between each example and the query. The ambiguous entry will take the sense where its example gives high similarity.

4.1. Using invers sense frequency (*isf*):

isf (inverse sense frequency) is equivalent to *idf* with little differences. We deals with all examples of one sense as one **block (like one document),** lets call it $\mathbf{s_j}$. Then for each word in the examples, *isf* is calculated by[3]:

$$isf_i = \log \frac{S}{n_i} \tag{6}$$

Where:

- S: is the number of senses for the orthogonal entry.
- n_i: is the number of senses where word wi appear in them examples.

The similarity between query and s_j, for the current ambiguous entry, is:

$$sim(\vec{q}, \vec{s}_j) = \frac{\sum_{w \in q,d} tf_{w,q} tf_{w,j} (isf_w)^2}{\sqrt{\sum_{i=1}^{n} (tf_{i,q} isf_i)^2} \times \sqrt{\sum_{i=1}^{m} (tf_{i,j} isf_i)^2}} \tag{7}$$

- $tf_{w,q}$, $tf_{i,q}$: are same as in equation 5.
- $tf_{w,j}$: term frequency of the word w in the examples of sense j.
- $tf_{i,j}$: is a term frequency of word i in the examples of sense j for current entry.
- isf_w: is inverse sense frequency of the word w.
- n and m are number of the different words in query and the examples of sense j respectively.

Using annotated corpus with the dictionary is very simple task with equation 7. Simply, we add all the sentences have specific entry of specific sense to the examples of this entry sense. For more details see the algorithm in figure 2.

[2] Orthogonal entry is the dictionary word has more than one sense. We used this term for differentiate it from the other words.
[3] P. Basile, etl. (2014) used IGF (Inverse Gloss Frequency)

5. Implementation of Lesk With Annotated Corpus And Results

The National Corpus of Polish project (Pol. Narodowy Korpus Jezyka Polskiego; NKJP) [14] was used as a dataset. The used senses inventory is NKJP_WSI (NKJP Word Sense Inventory). The whole corpus was taken with little exceptions[4]. This means, 844 annotated files were processed. The simplified Lesk algorithm is implemented on this corpus. The results, shown in table-1, were obtained by using training and testing 10 fold[5] of 844 files. The execution time for this experiment was very long time. In this implementation, we depended on the fact that the context was parsed previously and we got from this stage the ambiguous words and *all* their bases. Then, same test was used with deleting stop words. The results of this test are shown in table 2.

Table1. Applying simplified Lesk algorithm on NKJP corpus with deleting stop words.

fold	**#Files**		**matched senses**	**Total senses**	**Accuracy %**
	test	training			
1	84	760	1428	1792	79.687
2	84	760	505	628	80.414
3	84	760	723	850	85.059
4	84	760	661	775	85.29
5	84	760	176	203	86.699
6	84	760	333	409	81.418
7	84	760	796	954	83.438
8	84	760	1183	1449	81.642
9	84	760	1015	1206	84.162
10	88	760	992	1259	78.793
Total	**844**		**7812**	**9525**	**82.016**

Table2. Applying simplified Lesk algorithm on NKJP corpus with deleting stop words.

fold	**Files**		**matched senses**	**Total senses**	**Accuracy %**
	test	training			
1	84	760	1480	1792	82.589
2	84	760	533	628	84.872
3	84	760	720	850	84.705
4	84	760	668	775	86.193
5	84	760	179	203	88.177
6	84	760	356	409	87.041
7	84	760	816	954	85.534
8	84	760	1210	1449	83.505
9	84	760	1032	1206	85.572
10	88	760	1013	1259	80.460
Total	**844**		**8007**	**9525**	84.063

[4] Very little files have bags in its structure therefore we took 844 files.
[5] We did not use 10-Fold Cross-Validation. Simply we segmented the corpus to 10 parts (fold), One part as test and the other 9 parts as training.

6. CONSTRUCTION NEW DICTIONARY FROM THE CORPUS

Clearly, that using of annotated corpus with Lesk algorithm will increase the accuracy. But, practically, training need very long time for evaluating the terms. Certainly, it is used for first time but the performance can be improved by using a new dictionary from old dictionary which have all words (assist in deciding for solving WSD problem) for all senses. The word's *tf* and *isf* are recorded in the new dictionary with the word itself. Any word have zero *isf* will not be recorded. The algorithm for construction the dictionary is shown in figure 2. It will construct new dictionary automatically (we don't have query here).

By applying the algorithm, each word recorded in more than one sense will have the same *isf* and this is logically because *isf*=log(S/n_i). As mentioned below, this is very useful in practice.

tf for each word were recorded without using query and, logically, it will not affect the results when we will test this dictionary.

Algorithm for construction dictionary

1- start from(initialize the) existing dictionary "*NKJP_WSI.xml*"
2- Take one entry (ambiguous word w) from Dictionary
3- Get the senses and all words of the examples for each sense. Now, each sense has words list.
4- for the entry w: (i)search for this entry in the corpus (ii) all words of the sentence in which this entry appear are recorded in the sense words list of the matched sense.
5- For this entry of dictionary now we have list of senses where each sense have a list of words taken from the dictionary examples and the corpus. Compute *tf* and *isf* for each word where the 'documents' is the senses.
6- Delete each word have *isf*=0.
7- Record the words which are not deleted in new dictionary with its *tf* and *isf*.
8- Repeat 2-7 for all entries in the dictionary

Figure 2: Construction Dictionary from existing dictionary with assistant of annotated corpus

7. TESTING THE MODIFIED DICTIONARY

In practice we have query, sentence which have the ambiguous word w, but we don't have *isf* for each word in this sentence. This done easily by looking to the word if it is exist in any sense it will take the *isf* of the same word in the sense otherwise it's *isf*=0 (this because each word which recorded in more than one sense, it have same *isf*).

Computing *tf* for *w* in the query are done by comparing all the words in query with the *set* of all words in query without regards to the *set* of the words in document.

We used the same data set in section 5 for testing the new dictionary. The same two conditions, with/without taking stop words in computation, were taken.

We got on the results identical to the results in table 1 & 2. The important gain here is the time of execution where the time of execution drop down by 20 times comparing with the first tests.

8. ANALYZING THE RESULTS AND CONCLUSION

This paper presents that how to apply Lesk algorithm using dictionary with assistant of annotated corpus. Moreover, it presents how the similarity equation of IR is modified in order to apply it on WSD. Then, the performance (especially the time of execution) was improved by constructing a new dictionary from the annotated corpus and the used dictionary. The last modification is for practical use.

Firstly, all words in annotated corpus which related to the specific sense were obtained. Then, the set of all words for this sense and the other senses in same orthogonal entry where obtained which leads to words randomly ordered (or by frequency distribution). Finally, each word and its *tf* and *isf* are also obtained. This means statistical information about words is generated which shared in all senses for specific orthogonal entry.

We induced that, from practical test, there are a key words (limited words) which can be used to solve the WSD. The other words were used as assistant or weren't used in decision. As example, most of preposition weren't used for making the decision i.e. they can be taken as stop words (deleted). As mentioned, deleting the stop words increased the average accuracy on the tests.
the other thing was induced from this work, using dictionary alone for solving WSD problem not sufficient because some words depends on previous words not in the same sentence but may be in the previous sentences.

Anyone who has simple information about polish language, he/she knows that the base words in polish take much morphosyntatic forms. This will lead to collections of many words for the same base. From this point of view, this paper suggests writing dictionary depends on the bases. Furthermore, this work must be done carefully because it (maybe) will cause drop the accuracy. It need to more works and experiments for deciding if it can be used.

Because there are key words helped the algorithm in making decisions in WSD, searching about these words is important task. This work is purely related to linguistics scientists.

REFERENCES

[1] D. Jurafsky & J. Martin 2008 "Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition" PEARSON prentice Hall Upper Saddle River, New Jersey, USA.

[2] N. Ide and J. Véronis (1998). "Word Sense Disambiguation: The State of the Art" Computational Linguistics, Vol. 24 Issue 1, pp 2-40.

[3] H. Ng (1997) “Exemplar-Base Word Sense Disambiguation: Some Recent Improvements”. In Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing, EMNLP.

[4] A. Montoyo, A. Suarez, G. Rigau and M. Palomar (2005): “Combining Knowledge- and Corpus-based Word-Sense-Disambiguation Methods”. Journal of Artificial Intelligence Research Vol. 23, PP 299-330.

[5] Y. Ledo-Mezquita, G. Sidorov and V. Cubells, (2006) "Combined Lesk-based Method for Words Senses Disambiguation," In Proceedings of the 15 th International Conference on Computing, Washington: IEEE Computer Society.

[6] P. Basile, M. de Gemmis, A Gentile, P. Lops, and G. Semeraro. 2007. “UNIBA:JIGSAW algorithm for Word Sense Disambiguation”. In Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007), Prague, Czech Republic.

[7] S. Ponzetto and R. Navigli. 2010. “Knowledge-rich Word Sense Disambiguation Rivaling Supervised Systems”. In Proceedings of the 48th Annual Meeting of the Association for Computational linguistics,ACL ’10, Stroudsburg, PA, USA.

[8] F. Viveros-Jiménez, A. Gelbukh and G. Sidorov (2013) “Simple Window Selection Strategies for the Simplified Lesk Algorithm for Word Sense Disambiguation”.. Proceeding of 12th Mexican International Conference on Artificial Intelligence, MICAI 2013, Mexico City, Mexico.

[9] D. Schwab, A. Tchechmedjiev, J. Goulian, M. Nasiruddin, G. Serasset, and H. Blanchon 2013. “GETALP System : Propagation of a Lesk Measure through an Ant Colony Algorithm”. Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013), Atlanta, Georgia, USA.

[10] P. Basile, A. Caputo and G. Semeraro (2014) “An Enhanced Lesk Word Sense Disambiguation Algorithm through a Distributional Semantic Model”. Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics. Dublin, Ireland.

[11] R. Nvigli (2009) "Word Sense Disambiguation: A Survey" ACM Computing. Survey. Vol. 41, No.2, pp 1-69.

[12] S. Torres and A. Gelbukh (2009). "Comparing Similarity Measures for Original WSD Lesk Algorithm" Research in Computing Science 43, pp. 155-166

[13] A. Kilgarriff and J. Rosenzweig (2000). "Framework and results for English SENSEVAL." Computers and the Humanities, Vol. 34, pp 15- 48.

[14] A. Przepiórkowskiego, M Bańko, R. Górskiego B.Tomaszczyk (2012) “Narodowy Korpus Jenzka Polskiego” Wydawnictwo Naukowe PWN SA, Warsaw, Poland.

AUTHORS

Dr. Ahmed H. Aliwy is a lecturer in Computer Science Department in University of Technology, Baghdad, Iraq. His current research interests include Arabic Natural Language Processing ANLP, Machine Learning and Natural Language Processing. He has got his PhD from Warsaw University, Poland.

Dr. Ayad R. Abbas is a lecturer in Computer Science Department in University of Technology, Baghdad, Iraq. His current research interests include AI, Machine Learning and some applications of Natural Language Processing. He has got his PhD from University of Wuhan, China.