Ambiguity Spotting using WordNet Semantic Similarity in Support to Recommended Practice for Software Requirements Specifications

Jin MATSUOKA
IPS, Waseda University
Kitakyushu, Japan
jinmatsuoka@akane.waseda.jp

Yves LEPAGE
IPS, Waseda University
Kitakyushu, Japan
yves.lepage@aoni.waseda.jp

***Abstract*—Word Sense Disambiguation is a crucial problem in documents whose purpose is to serve as specifications for automatic systems. The combination of different techniques of Natural Language Processing can help in this task. In this paper, we show how to detect ambiguous terms in Software Requirements Specifications. And we propose a computer-aided method that signals the reader for possibly ambiguous usage of terms. The method uses compound term measure (C-value), WordNet semantic similarity (WordNet wup_similarity) and a proposed semantic similarity measure between sentences.**

Keywords-SRS, WordNet, Semantic similarity, Clustering, C-value, Ambiguity spotting

Ⅰ. INTRODUCTION AND BACKGROUND

In the manufacturing industry, QCD is an acronym for Quality, Cost and Delivery for methods that use metrics on these three factors to ensure low cost, high quality and on-time delivery in the production of goods. Software being a product as any other one, similar methods have spread into the software development industry, leading, in particular, to the adoption of software metrics to ensure the quality of programs (from number of lines of codes per function to cyclomatic complexity or the like).

As software development does not reduce anymore to programming, the last decades have also seen radical improvements for upstream software design with the introduction of different methods (MERISE, VDM, Z notation, object- or aspect- oriented paradigms), together with a modeling languages like VDM-SL or UML.

As the very first step in software design remains requirements, the concern about quality upstream has led to a certain number of recommendations like IEEE std 830[1] for Software Requirements Specifications (SRS) themselves. SRS are texts written in natural language that mainly describe the functionalities of the software to deliver. Among other problems, the above-mentioned set of recommendations pinpoints contradictions or ambiguities, and the problem of their early detection.

Contradictions can be addressed using formal methods relying on ontologies and/or description logic, for which tools like Alloy [3], spin, LTSA exist. The methods are using first order predicate logic or extended first order predicate logic. However, such methods are usually heavy to implement and generally require the involvement of experts. Also, in the filed of Natural Language Processing (NLP), they try to solve the problem with deep (semantic) parsing.

This paper is concerned with the problem of the detection of word ambiguity in SRS as a means to contribute to software development quality, and pleads for computation-light resource-based techniques. In Natural Language Processing (NLP), the problem of Word Sense Desambiguation (WSD) has been addressed in the framework of Information Retrieval/Extraction (IR/IE) [4], [5], [6] or Machine Translation (MT) [7], [8] with a different perspective from ours. Research close to our perspective is [1], [2], the difference being that our approach relies on similarity computation in a hierarchical semantic network. Our goal is to design a checker that will run on SRS texts and spot possibly ambiguous words or terms inside sentences. The purpose is to help a human reader to make a decision about the correct use of the word or term.

The rest of the paper is organized as follows: Section Ⅱ introduces the resource that we propose to use, and the NLP techniques at the core of our proposed method. Section Ⅲ gives an overview of the proposed method and presents the processing flow. Section Ⅳ presents an experiment on a real SRS text and reports some measures so as to evaluate our method.

Ⅱ. THE NLP RESOURCE AND TECHNIQUES USED

We introduce a resource and three techniques for detecting ambiguous terms. Two techniques of similarity are like a kind of knowledge based method and another is linguistic and mathematical method.

A. The WordNet semantic network

The WordNet semantic network is a hierarchical semantic view of the vocabulary of a language. Other name is *Lexical Ontology*. The first one was created at the University of Princeton for English, and there now exists such semantic networks for a range of languages, among which, Japanese[2].

1 http://standards.ieee.org/findstds/standard/830- 1998.html

2 http://nlpwww.nict.go.jp/wn- ja/index.en.html/S.

The nodes in a WordNet are sets of synonyms (*synsets* for short) that stand for a concept. An ambiguous word, i.e., a word with more than two different meanings, will thus belong to two different synsets. For instance, the word /ika/ in Japanese is ambiguous, as it can be found in two different synsets in the Japanese WordNet. The two meanings are: 'a toy consisting of a light frame covered with tissue paper, a kite' and 'cooked squid'[3]. In the context of a sentence, it is usually possible to decide the meaning of 'ika' from the other words in the sentence:

Sentence	Meaning	
	'kite'	'cooked squid'
We raise an /ika/ on the hill.	yes	no
We fry an /ika/ in oil.	no	yes

Links between synsets describe different semantic relations like holonymy/meronymy (whole/part), antonymy, etc. In particular, the relationships of hypernymy/hyponymy define a hierarchical structure between syntsets. The hierarchy has a top node (concept), and thus allows for the computation of semantic similarity.

B. *Semantic similarity in WordNet*

As the hierarchy of WordNet has a top node (concept), any two synsets can be assigned a semantic similarity measure using the hierarchy. The Wu and Palmer similarity metric uses the depths of two given concepts (synsets) in the WordNet taxonomy. There are several similarity metrics for WordNet like Lin similarity [10], Lesk similarity [11] or Wu and Palmer similarity [9]. For any synset, the length of the path from the top of the hierarchy to the synset is noted as *depth(s)*. Now, for two synsets s_1 and s_2, it is possible to find their lowest common synset by going up towards the top of the hierarchy; this synset is denoted as $lcs(s_1,s_2)$. The semantic similarity of s_1 and s_2 is defined using the common node (concept) of the paths of the two synsets to the top of the hierarchy. Using a formula *a la* Dice coefficient, one defines:

$$wup\_similarity(s_1,s_2) = \frac{2 \times depth(lcs(s_1,s_2))}{depth(s_1) + depth(s_2)}$$

Figure 1 illustrates the above equation and it's computation.

As a word may belong to several synsets, the semantic similarity between two words is defined as the smallest similarity between all the possible synsets the two words at hand belong to. In addition, equality of part of speech is required.

$$word\_sim(w_1,w_2) = \max_{(s_i,s_j)} wup\_similarity(s_i,s_j)$$

this maximum being taken over all (s_i,s_j) such that:

$$w_1 \in s_i \wedge w_2 \in s_j \wedge pos(w_1) == pos(w_2)$$

3 The Japanese WordNet misses a third basic meaning: 'A squid (animal)'.

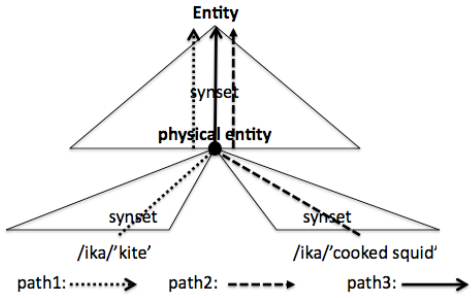

Fig. 1. Computation of the semantic similarity between the two synsets for the same words /ika/: 'kite' and 'cooked squid'. The figure shows the two synsets and their paths to the top of the hierarchy in WordNet. The least common concept is in noted as *physical entity*. The value of the similarity is $2 \times \text{length(path3)} / (\text{length(path1)} + \text{length(path2)})$.

where *POS* is *part of speech*. In the absolute, a term is ambiguous if it belongs to several synsets. This is the case of our example word /ika/. However, when used in a sentence, a word has usually one meaning. This meaning is constrained by the other words in the sentence. The disambiguation process relies on the fact that the other words in the sentence are closer to one of the synsets the ambiguous word belongs to. This synset is naturally preferred for the interpretation of this word in the sentence and thus becomes the meaning of this word in the sentence. To simulate this disambiguation process, we thus compute the similarity of each word of a sentence with the ambiguous word. The intuition would be that the average of the similarities of each word in the sentence with the ambiguous word would lead better disambiguation. Our experiments showed that taking the max over the words in the sentence delivers better results.

Let a word w belong to the synset s, and let this word appear in a sentence S. We define the semantic value of the synset for the w in the sentence S in the following way:

$$sem\_val(s,S) = \max_{s_j / w_i \in s_j \wedge w_i \in S - \{w\}} word\_sim(s,w_i)$$

This is applied to disambiguate a word w with several synsets s_i by taking the synset which exhibits the highest semantic value. In other words, the meaning s (a synset) of an ambiguous word w in a sentence S is defined as:

$$s = \arg\max_{s_i / w \in s_i} sem\_val(s_i,S)$$

To illustrate with an example, let us see the result obtained for the sentence: *we raise a /ika/ in the park.*

	We	raise	an	/ika/	in	the	park
'kite'	0.00	-	-	-	-	-	**0.56**
'cooked squid'	0.00	-	-	-	-	-	0.22

In the general case, several synsets may get the same max value, so that *argmax* may be a set with several elements, which is a case of ambiguity. On the following sentences that all contain the word /ika/, the previous definition leads to the results shown on the right column of the following table. These results clearly show that disambiguation is performed rightly in the obvious cases, while the word /ika/ remains ambiguous in more subtle cases, meeting human intuition.

Sentences	Computed meaning
We raise an /ika/ in the park.	{'kite'}
We raise an /ika/ on the hill.	{'kite','cooked squid'}
We fry an /ika/ in oil.	{'cooked squid'}
The diner is an /ika/.	{'cooked squid'}
We boil an /ika/ in water.	{'kite', 'cooked squid'}
We eat an /ika/.	{'kite', 'cooked squid'}

This shows that the context of only one sentence is not enough to disambiguate a term in many cases. The details for the computation on the second sentence are given in the following table.

	We	raise	an	/ika/	on	the	hill
'kite'	0.00	-	-	-	-	-	**0.13**
'cooked squid'	0.00	-	-	-	-	-	**0.13**

The previous results show that the sentence context only is not enough to perform disambiguation, and that, although two sentences may look very similar, disambiguation can be performed in one case (sentence with park), whilst not in the other case (sentence with hill).

C. Sentences clustering by semantic meanings

The previous results can be interpreted as a clustering process by which the sentences are clustered in clusters according to the set of meanings (synsets) taken by the ambiguous word.

Cluster	Sentences
{'kite'}	We raise an /ika/ in the park.
{'cooked squid'}	We fry an /ika/ in oil. The diner is an /ika/
{'kite','cooked squid'}	We raise an /ika/ on the hill. We boil an /ika/ in water. We eat an /ika/.

The process of clustering can be carried on by exploiting these clusters. As in partitional clustering, sentences where the term remains ambiguous can be checked for their similarity to each unambiguous cluster. If the similarity to a cluster is found to be enough large, the meaning of the term in the ambiguous sentence can be considered the one in the non-ambiguous cluster, thus leading to disambiguation. In the case no reliable similarity can be computed, disambiguation cannot be performed.

The Japanese WordNet misses a third basic meaning: 'A squid (animal)'.

D. Semantic similarity between sentences

In order to evaluate the similarity between two sentences S_1 and S_2, we use the two possible directional semantic alignments between them. A directional alignment gives the best corresponding word in the second sentence for each word of a first sentence.

	we	boil	an	/ika/	in	water
we	**1.0**	-	-	-	-	-
raise	-	-	-	-	-	-
an	-	-	**1.0**	-	-	-
/ika/	0.0	-	-	**1.0**	-	0.7
on	-	-	-	-	-	-
the	-	-	-	-	-	-
hill	0.0	-	-	0.1	-	**0.5**

The sum of the best semantic similarities over all words of one sentence partly reflects the semantic similarity between both sentences. Notice that this similarity is not symmetrical as the number of words in the two sentences may be different. For this reason, we average over both directions.

In order to reflect the importance of words, i.e., to give more weight to words that are characteristic of the sentences over the words which are very frequent in a language, we give a weight to each word by using the idf measure ordinarily used in document retrieval [15]. This is similar to what is done in [13], [14].

To summarize, we propose to compute the similarity between two sentences S_1 and S_2 according to the following formulae:

$$sent\_sim(S_1,S_2) = \frac{1}{2} \times (sem\_div(S_1,S_2) + sem\_div(S_2,S_1))$$

$$sent\_div(S_1,S_2) = \frac{\sum_{w_i \in S_1} (\max_{w_j \in S_2} word\_sim(w_i,w_j) \times idf(w_i))}{\sum_{w_i \in S_1} idf(w_i)}$$

$$idf(w) = -\log \frac{|S \in D / w \in S|}{|D|}$$

where D is the document and $|D|$ is the number of sentences in the document.

E. Partitional clustering of sentences

To be incorporated in a cluster, the similarity of the sentence to the cluster should be enough large, based on the results of experiments, we fixed a threshold of **0.7** for this.

With the previous sentence similarity and the threshold mentioned above, our example sentences are now clustered in the following way:

Cluster	Sentences
{'kite'}	We raise an /ika/ in the park. **We raise an /ika/ on the hill.** **We boil an /ika/ in water.**
{'cooked squid'}	We fry an /ika/ in oil. The diner is an /ika/.
{'kite','cooked squid'}	We eat an /ika/.

As a result, one sentence only remains where the word /ika/ is still ambiguous. For other sentences, the word is seemingly not ambiguous.

This kind of configuration is not desired for SRS. Indeed, the opposite extreme case is searched for a term used in a SRS: all sentences should use the term in the same unique meaning. In the next section, we turn to the problem of spotting the word for which ambiguity should be checked.

F. Terms detection

According to IEEE std 830, the words for which ambiguity should be checked are those words that are relevant for the requirements at hand, i.e., technical terms. To extract technical terms or compound terms from technical documents (among which, SRS), a method called *C-value* method has been introduced in [12]. The C-value of a term is computed as follows:

$$C-value(a)=\begin{cases}\log_2|a|\times\left(f(a)-\dfrac{1}{|T(a)|}\sum_{b\in T(a)}f(b)\right)\\ \log_2|a|\times f(a)\quad if\quad T(a)=\phi\end{cases}$$

where *a* is a candidate term, |*a*| its lengths in words and *f(a)* its frequency. *T(a)* represents the set of candidate terms that contain *a* and |*T(a)*| stands the cardinal of this set, i.e., the number of other candidate terms containing *a*. The second line of the definition gives the formula for terms that are not included in any other candidate term and candidate term *b* is contained in *a*.

The method combines linguistic and statistical information. And the method enhances the common statistical measure of frequency of occurrence for term extraction, making it sensitive to a particular type of multi-terms, the nested terms.

For each term extracted by C-value method computation, we check for its presence in the WordNet semantic network. It a term is not found, we decide that each of its word becomes a term to check for ambiguity.

III. Overview of the Disambiguation Method

To summarize in a formalized way, the different steps of our method can be described as follows. Let *T* be the text we want to check for ambiguous use of terms (a SRS text in our experiment below). *T* is considered as a set of sentences. Each sentence is considered as a set of words. Let w be a candidate term in this text selected by use of C-value computation. In a first step, we gather the set of all meanings from WordNet for the word at hand, w. In a second step, we assign the set of possible meanings of w in *S* to each sentence *S* using the semantic value function define in Subsection II-B. In a third step, we reassign the set of meanings of w in *S* by use of partitioning clustering as described in Section II-E that is, we enforce disambiguation when possible. To do this, we use the semantic similarity between sentences defined in Subsection II-D. The final step consists in clustering the sentences by sets of meaning.

$\Sigma = \{\sigma \in \text{WordNet}/w \in \sigma\}$ // all meanings of w
S = $\{s \in T/\ w \in s\}$ // all sentences containing w
for $s \in S$ **do**
  $m(s) = \{\sigma \in \Sigma/\sigma = \text{sem\_val}(w, s)\}$
end for
for $s \in S$ such that $|m(s)| > 1$ **do**
  $s'' = \text{argmax}_{s' \in S}\text{sent\_div}(s, s')$
  if $\text{sent\_sim}(s, s'') > \theta$ and $m(s'')$ is a singleton **then**
    $m(s) = m(s')$
  end if
end for
partition S according to the values of m

IV. Experiment and Results

We performed an experiment on an actual SRS text, in the Japanese language, in order to evaluate the possibilities of our proposed method of disambiguation.

A. Experimental Data and Processing Chain

The SRS relates to an 'Interior Lighting Power Saving System'. This file was delivered to us in Word format. A processing chain consisting of the following steps was used to process the text:

- extract plain text from Word format (by use of antiword[4]);
- separate into paragraphs and sentences (by use of in-house software);
- apply Japanese word segmentation to each sentence (by use of JUMAN[5]);
- collect terms (by use of in-house implementation of C-value computation) and possibly split compound term into single word term (after checking for presence in WordNet);
- for each term, for each sentence which contains the term, check whether the term is ambiguous in this sentence;
- perform clustering of sentences by sets of meanings;
- for each term, report the term when more than one meaning is detected in the text. In this case, report the clusters and those signal sentences where the word is still found ambiguous.

4 http://www.winfield.demon.nl/
5 http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?JUMAN

TABLE I
TERMS EXTRACTED BY C-VALUE COMPUTATION (THE RIGHTMOST COLUMN INDICATES THE PRESENCE IN WORDNET)

Extracted term	Meanings	C-value	in WordNet?
/syoumei souti/	'lighting device'	86.1	yes
/nettowaaku seigyo souti/	'network control device'	69.7	no
/nettowaaku seigyo souti musen mozyuuru/	'network control device radio module'	30.2	no
/souti sisutemu parametaa houti messeeji/	'device system parameter alarm message'	27.3	no
/souti sisutemu parametaa/	'device system parameter'	26.9	no
/souti radio mozyuuru/	'device radio module'	24.8	no
/souti touroku youkyuu message/	'device registration requirements message'	18.7	no
/syoumei souti sistemu parametaa houti messeezi/	'lighting device system parameter alarm message'	18.1	no
/syoumei souti sistemu parametaa/	'lighting device system parameter'	16.7	no
/messeezi zentai/	'entire message'	14.0	no

Many of the sentences in the text contain special characters. Also, since this SRS describes technical details in physical terms, there are many formulae. Our system automatically eliminates formulae, tables and arrays by checking the proportion of characters that do not belong to the standard Japanese writing system (kanji, hiragana and katakana). The number of remaining sentences was 1,177. The following table shows statistics about the kind of characters found in the SRS text used in our experiments:

feature	the number of feature
kanji	2832
hiragana	1458
katakana	2535
digit	950
alphabet	2759
comma	104
period	118
special character	4006
total	14762

The terms extracted by the system from these sentences, are given in Table Ⅰ. Obviously, very long terms are unlikely to be found in WordNet. As a result, almost all the single words entering in these compound terms were checked for ambiguity by our system. This resulted in the 17 terms listed in Table Ⅱ.

B. Results and Evaluation

The results of the experiment are reported in Table Ⅱ. The IEEE std 830 recommendation for SRS is that a term should have only one meaning in an SRS. Here, this means one cluster with only one meaning. On the 17 terms examined automatically, there were 6 such IEEE std 830 compliant terms (6/17 ≃ one third). For /syoumei souti/ 'lighting device', the number of meanings in WordNet is 1, the number of sentences is 48, adapted clusters is 1.0 by our computer aided method and the number of meanings per clusters is 10. That is, according to IEEE std 830, this term is no problem.

Two thirds of the terms can be signaled as non IEEE std 830 compliant. No cases of one cluster with several meanings was observed. A particular case is the case of the term meaning `control', used in 10 sentences that can be interpreted in 5 different separate ways according to our results (5 clusters of only one meaning each). When inspecting the definition of this word in WordNet and its different synsets, it was judged difficult to make a clear separation between them.

In all the remaining cases, each term was classified into several clusters, each cluster being itself ambiguous (average meaning per cluster >1).

TABLE II
FINAL RESULT OF THE DISAMBIGUATION PROCESS FOR EACH TERM.

Term	Translation	meanings (= synsets) in WordNet	# of sentences	# of clusters	# of meanings per clusters	IEEE 830 compliant
/syoumei souti/	'lighting device'	1	48	1	1.0	ok
/nettowaaku/	'network'	4	39	2	2.0	-
/seigyo/	'control'	10	46	5	1.0	-
/souti/	'device'	6	74	2	2.7	-
/musen/	'radio'	4	40	1	1.0	ok
/mozyuuru/	'module'	3	19	1	1.0	-
/sisutemu/	'system'	8	23	2	1.5	-
/parametaa/	'parameter'	4	16	3	1.3	-
/touroku/	'registration'	4	10	1	1.0	ok
/youkyuu/	'requirements'	20	22	4	1.3	-
/messeezi/	'message'	4	42	2	2.0	-
/sistemu/	'system'	8	27	4	1.3	-
/houti/	'alarm'	2	14	1	1.0	ok
/syoumei/	'lighting'	5	50	4	1.3	-
/zentai/	'entire'	5	29	1	1.0	ok
/daata/	'data'	3	12	1	1.0	ok
/seetei/	'establishment'	9	24	4	2.5	-

Ⅴ. CONCLUSION

In this paper, we designed and tested a word sense disambiguation method that uses different techniques from Natural Language Processing so as to automatically check the conformity of software requirements specifications (SRS) to recommendations contained in IEEE std 830. The techniques used consist in distance similarity in WordNet, inverse document frequency (idf) to measure the importance of words, and C-value to extract technical terms. The method is supposed to provide a human reader with signals on possibly ambiguous terms.

In an experiment on an actual SRS, we showed that two thirds of the terms used were judged ambiguous by the proposed automatic method.

REFERENCES

[1] Erik Kamsties, Daniel M. Berry, Barbara Paech, *Detecting Ambiguities in Requirements Documents Using Inspections*, Proceedings of the First Workshop on Inspection in Software Engineering WISE0, 2001.

[2] Hui Yang, Alistair Willis, Anne De Roeck, Bashar Nuseibeh, *Automatic Detection of Nocuous Coordination Ambiguities in Natural Language Requirements*, ASE'10, 2010.

[3] Daniel Jackson, *Software Abstractions: Logic, Language, Analysis*, The MIT Press, 2006.

[4] Dumais, Susan T, *Enhancing performance in latent semantic index-ing (LSI) retrieval*, Technical Re-port TM-ARH-017527, Bellcore, Morristown, NJ,1990.

[5] Thomas Hofmann, *Probabilistic Latent Semantic Indexing*, Proceedings of the Twenty Second Annual International SIGIR Conference on Research and Development in Information Retrieval, 1990.

[6] David M. Blei, Andrew Y. Ng, Michael I. Jordan, *Latent Dirichlet Allocation*, Journal of Machine Learning Research, 2003.

[7] Marine Carpuat, Dekai Wu, *Improving Statistical Machine Translation using Word Sense Disambiguation*, Association for Computational Lingusitics, 2007.

[8] Jason Adams, *Word Sense Disambiguation and Machine Translation*, IJCNLP, 2005

[9] Wu and M. Palmer, *Verb semantics and lexical selection*, In Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics, 1994.

[10] D. Lin, *An information theoretic definition of similarity*, In Proceedings of the 15th International Conference on Machine Learning, 1998.

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

[12] Katerina Frantzi, Sophia Ananadou, Hideki Mima, *Automatic Recognition of Multi-Word Terms: the C-value/NC-value Method*, International Journal on Digital Libraries, 2000

[13] Courtney Corley and Rada Mihalcea, *Measuring the Semantic Similarity of Texts*, EMSEE '05 Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment, 2009

[14] Thanh Ngoc Dao, Troy Simpson, *Measuring Similarity between sentences*, http://googlecode.com, 2002

[15] G. Salton and M.McGill, *Introduction to modern information retrieval*, McGraw-Hill New York, 1983.