

An Innovative Two-Stage WSD Unsupervised Method

Un Innovador Método No Supervisado para Desambiguación de Sentidos de Palabras basado en dos etapas

Javier Tejada Cárcamo
Centro de Investigación en Computación.
Instituto Politécnico Nacional.
Ciudad de México, 07738, México
Sociedad Peruana de Computación, Perú.

jawitejada@hotmail.com

Alexander Gelbukh, Hiram Calvo
Centro de Investigación en Computación.
Instituto Politécnico Nacional.
Ciudad de México, 07738, México

gelbukh@gelbukh.com, hcalvo@cic.ipn.mx

Abstract. An unsupervised method for word sense disambiguation is proposed. The sense of the word is chosen to be the most similar to the senses of other words that appear in the corpus in similar contexts. Training consists of building a weighted list of related words (*quasi-synonyms*) for each word; the weights are obtained by measuring similarity between the word's contexts. We adapt the algorithm of McCarthy *et al.* 2004 for finding the best sense in each occurrence, instead of finding the predominant sense of each word in the entire corpus. Their maximization algorithm allows then each quasi-synonym to accumulate a score for each ambiguous word sense; the sense with the highest score is chosen. We obtain a top precision of 69.86% using the same corpus for training and disambiguating.

Keywords: Natural Language Processing, Unsupervised Machine Learning, Word Sense Disambiguation, Semantic Similarity.

Resumen: Se propone un método no supervisado para la desambiguación de sentidos de palabra. El sentido de un vocablo ambiguo depende de los sentidos de otras palabras que aparecen en contextos similares en un corpus. El entrenamiento consiste en obtener una lista ponderada de sinónimos o palabras relacionadas (*quasi-sinónimos*) para cada vocablo del corpus tomando en cuenta la similitud de sus contextos. Adaptamos el algoritmo de McCarthy *et al.* 2004 para encontrar el mejor sentido de cada ocurrencia, en lugar de encontrar el sentido predominante de cada palabra en todo el corpus. Su algoritmo de maximización permite entonces que cada *quasi-sinónimo* acumule puntaje para cada sentido del vocablo ambiguo. El sentido con puntaje más alto es el seleccionado. Se obtuvo una precisión máxima de 69.86% usando el mismo corpus para entrenamiento y desambiguación.

Palabras clave: Procesamiento de Lenguaje Natural, Aprendizaje no Supervisado, Desambiguación de Sentidos de palabras, Similitud Semántica.

1 Introduction

Word sense disambiguation (WSD) consists of determining the sense expressed by an ambiguous word in a specific context. For example, for *doctor* there are three senses listed in WordNet: (1) a person who practices medicine, (2) a person who holds a Ph.D.

degree from an academic institution; and (3) a title conferred on 33 saints who distinguished themselves through the orthodoxy of their theological teaching. In order to find the right structure from a text the right sense of a particular word must be chosen.

There are two different types of methods to approach this problem: supervised and unsupervised. Supervised methods consist of

classifiers which associate a specific context to each sense of the ambiguous word using manually tagged corpora. In this way, supervised methods determine the sense of future ambiguous instances of a word. This process is costly mainly in building a tagged corpora: a large quantity of annotated text is needed for a good performance. If there are not enough ambiguous word instances, the problem of *data sparseness* arises. Many unsupervised methods try to build these classifiers automatically, obtaining contexts by sense. Other methods, based on the assumption that *different words have similar meanings if they are presented in similar contexts*, try to get contexts by word. (Pedersen *et al.*, 2004). These contexts are used in later stages of clustering and word sense discrimination techniques. (Leacock, C. and M. Chodorow. 1998)

In our method, we obtain a list of synonyms or related words (quasi-synonyms) for each ambiguous word. That is, other words that are used in contexts similar to those surrounding the ambiguous word, within a specific corpus. These *quasi-synonyms* will determine the sense for a word using the maximization algorithm presented in (McCarthy *et al.* 2004). This algorithm allows each quasi-synonym to accumulate a score for each sense of the ambiguous word, so that the sense which has the highest score is chosen.

The main contribution of this work is the method of obtaining quasi-synonyms. For this purpose we collect all the contexts in a corpus where a specific word is present, and then we use this information to build a semantic similarity model that measures the semantic distance between the words of the training corpus. The *quasi-synonyms* of an ambiguous word are those which are the closest by their contexts.

Quasi-synonyms of any word change dynamically depending on their local contexts and the corpus. For example, in *The doctor cured my wounds with a medicine*, the *quasi-synonyms* for *doctor* would be: *physician, medicine, alcohol, lint*; however, in *The doctor published his latest research in the conference*, the *quasi-synonyms* of *doctor* would be *scientific, academic, university, conference*.

Originally, the maximizing algorithm proposed in (McCarthy *et al.* 2004) was used to obtain the predominant sense of a word. In their work, the context for the ambiguous word is not considered: Its *quasi-synonyms* are obtained

from Lin's Thesaurus (Lin, D. 1998). In *The stars of the sky are brighter in the coastline*, the top 5 *quasi-synonyms* from the Lin's thesaurus for the word *star* are: *fame, glamour, money, Hollywood, constellation*. We can see here that these *quasi-synonyms* reflect poorly the sense of heavenly body.

We will describe further details of our method in the following sections. Section 2 describes the training stage; Section 3 describes the disambiguation stage. Section 4 describes our experiments. Finally, we conclude in Section 5.

2 Training Stage

Training consists of creating a semantic similarity model for each corpus to be disambiguated. The model was built as a Word Space Model (WSM) (Karlgrén, J. and M. Sahlgrén. 2001), which determines the proximity or semantic distance between the words of a corpus. First we obtained the contexts in which each word is presented in a particular corpus. This information was then organized in our WSM. (Schütze, H. 1993).

2.1 Obtaining Contexts

The first step in building a semantic similarity model is to collect all the contexts for each word in a corpus. Among the definitions of context, we have chosen *syntactic context*. We used MINIPAR syntactic analyzer presented in (Lin, D. 1998), to obtain dependency relationships in a corpus. Dependency relationships are binary asymmetric relationships between a *head word* and a *modifier word*. These dependency relationships build a tree that connects all the words in a sentence (Allen, J. 2000). A *head* may have several *modifiers*, but each modifier has only one *head*. (Mel'čuk, Igor A. 1987).

Once we have a tree, we apply further transformations to filter out less useful relationships: Ignore prepositions – see Figure 1 and Include sub-modifiers as modifiers of the head – see Figure 2.

We obtain syntactic modifier dependencies for each word in the corpus. See formula (1)

$$L(word_n) = \{(mod_1, f_1), \dots, (mod_n, f_n)\} \quad (1)$$

where $word_n$ is a word in the corpus, mod_n is a syntactic modifier of $word_n$, and f_n is the

frequency of mod_n and $word_n$ appearing together.

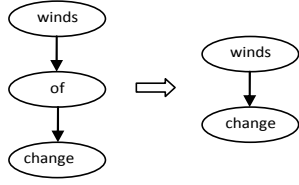


Figure 1: Ignoring prepositions

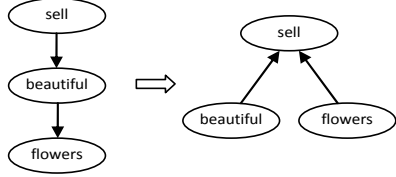


Figure 2: Sub-modifiers as modifiers of a head

2.2 Semantic Similarity Model

Once the syntactic context for each word has been obtained, we use a Word Space Model (WSM) to represent the information to be used for measuring semantic similarity. WSM is a spatial representation of word meaning. The main idea behind WSMs is that semantic similarity can be represented as proximity in an n -dimensional space, where n can be any integer ranging from 1 to some very large number.

This term is due to (Schütze, H. 1993), who defines it as follows: Vector similarity is the only information present in Word Space: semantically related words are close, unrelated words are distant. WSM is based on the geometric metaphor of meaning proposed in (Lakoff, G., and M. Johnson. 1980), (Lakoff, G., and M. Johnson. 1999) and the distributional hypothesis. (Schütze, H., and J. Pedersen. 1993) argue that meanings are locations in a semantic space, and semantic similarity is the proximity between those locations and (Sahlgren, Magnus. 2006) argue that words with similar distributional properties have similar meanings.

Implementation of WSM is based on the model of vector space, also known as the TF-IDF scheme (*term frequency - inverse document frequency*). This model is usually used for classification tasks and for measuring document similarity. Each document is represented by a vector whose number of dimensions is equal to the quantity of different words that are in it.

In our method, the number of dimensions of the WSM is the same as the number of different words in the corpus. Each word is represented by a vector and the *word's modifiers* determine the weight w in each dimension. This value is calculated as the product of TF and IDF corresponding to that *modifier*. The weight represents the affinity degree between a *word* and a *modifier* when they are represented in the model. TF reflects the importance of a *modifier* with regard to the *word* that it is modifying. Its value is greater if the *modifier* appears more often with that *word*. IDF measures the importance of a *modifier* with respect to the remaining *words* in the same corpus. The weight of a *modifier* decreases if it appears more often with other *heads* of the corpus, and it increases when it appears with a smaller number of *words*. This is because highly frequent *modifiers* have a low factor of discrimination when *words* are represented by a vector (Schütze, H., and J. Pedersen. 1993). Formulas 2, 3, y 4 show these measures.

$$f_{i,j} = \frac{freq_{i,j}}{\max freq_{1,j}} \quad (2)$$

$$idf_i = \log \frac{N}{n_i} \quad (3)$$

$$w_i = f_{i,j} \times idf_i \quad (4)$$

Where $freq_{i,j}$ is the frequency of the *modifier* _{i} with *word* _{j} , $\max freq_{1,j}$ is the highest frequency of the modifiers of *word* _{j} , N is the number of *words* in the corpus, n_i is the number of *words* which *modifier* _{i} modifies, and w_i is the final weight.

The weights w calculated for all *modifiers* of each *word* are represented as a vector in our WSM. See formula 5.

$$V(word_i) = \{(dim_1, w_1), \dots, (dim_n, w_n)\} \quad (5)$$

Where $V(word_i)$ is the vector which represents *word* _{i} , n is the number of dimensions of our WSM, dim_n is each dimension of the WSM (there are as many dimensions as there are different *words* in the corpus), and w_n is the weight assigned to dim_n . Several dimensions for a *word* are weighted as 0 because the *modifier* corresponding to that dimension was not found related to this *word*.

3 Disambiguation stage

In this stage we describe how the sense of an ambiguous word is obtained, considering its syntactic context, the created word space model, and the maximization algorithm proposed in (McCarthy *et al.* 2004). In that work, McCarthy *et al.* propose obtaining the predominant sense in a word for the overall corpus, while we adapt their algorithm to a local context, finding a different sense for each context.

3.1 Obtaining Quasi-Synonyms

One of the premises of the *context similarity* concept can be stated as: *two different words are semantically related if they are presented in similar contexts*. Based on this premise, we try to find terms which are used in contexts similar to those of the ambiguous word. We call these terms *quasi-synonyms*. These terms vary depending on the syntactic context of the word and the corpus from which the WSM has been created, as Figure 3 shows.

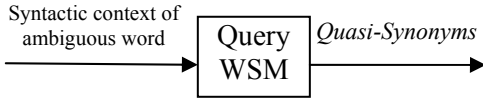


Figure 3: Obtaining Quasi-Synonyms

Extraction of *Quasi-synonyms* consists of the following steps: First, we extract the syntactic dependencies for each ambiguous word and then we create a query vector with the same number of dimensions as the WSM. This vector is compared with each of the vectors which represent the words of the corpus in the WSM. There are several ways to compute the similarity between two vectors (Patwardhan *et al.*, 2003). In our method this is determined by the cosine value of the angle measured between such vectors (Formula 6).

$$\text{Cosine_value} = \frac{\vec{v}_j \bullet \vec{q}}{\left| \vec{v}_j \right| \times \left| \vec{q} \right|} \quad (6)$$

Where \vec{v}_j is the vector that represents each word in the corpus and \vec{q}_j is the query vector which represents the syntactic context of the ambiguous word. After comparing \vec{q}_j with every other word of the WSM, we obtain a

weighted list of *quasi-synonyms* represented in Formula 7.

$$QS(word_i) = \{(qs_1, w_1), \dots, (qs_n, w_n)\} \quad (7)$$

Where qs_l is the quasi-synonym that is the most semantically related to the ambiguous word $word_i$ and qs_n is the quasi-synonym that is the least related to $word_i$. w_n is the weight of qs_n .

3.2 Choosing the right sense

Once the quasi-synonym list has been obtained, we use the maximization algorithm proposed in (McCarthy *et al.* 2004) to label syntactically the ambiguous word. This algorithm allows each quasi-synonym to accumulate a score for each sense of the polysemous word. The sense with the highest score is selected. Formulas 8, 9 and 10 show how the quasi-synonym list accumulates a score for a sense. See also Figure 4.

$$\text{Weight}(w_{si}) = \sum_{qs_j \in QS_w} P(w, qs_j) \times P_{Norm}(w_{si}) \quad (8)$$

$$P_{Norm}(w_{si}) = \frac{pswn(w_{si}, qs_j)}{\sum_{w_{si} \in sentidos(w)} pswn(w_{si}, qs_j)} \quad (9)$$

$$pswn(w_{si}, qs_j) = \max_{s_x \in senses(qs_j)} (pswn(w_{si}, s_x)) \quad (10)$$

In this equation, w is the ambiguous word, w_{si} is each one of the senses of w , QS_w is the set of *quasi-synonyms* of w , and qs_j is each quasi-synonym. $P(w, qs_j)$ represents the semantic similarity between w and qs_j . This value has been computed in the WSM. P_{Norm} represents how we normalize the weight of w_{si} using all the senses of w and the current qs_j .

The function $pswn$ returns the sense of a word that has the greatest semantic similarity to a particular sense. For example, $pswn(w_{si}, qs_j)$ compares all the senses of the quasi-synonym qs_j with w_{si} and obtains the sense of qs_j which has more semantic similarity with regard to w_{si} .

We use WordNet::Similarity presented in (Patwardhan *et al.*, 2003) to measure semantic similarity between two senses. This is a set of libraries that implement similarity and semantic relationship measures in WordNet (Miller, G., 1990)¹. Following (McCarthy *et al.* 2004), we used Jiang–Conrath (JCN) measure.

¹ These measures were proposed in (Resnik, P. 1995), (Lin, D. 1998), (Jiang, J. and D. Conrath. 1997) and (Leacock, C. and M. Chodorow. 1998).

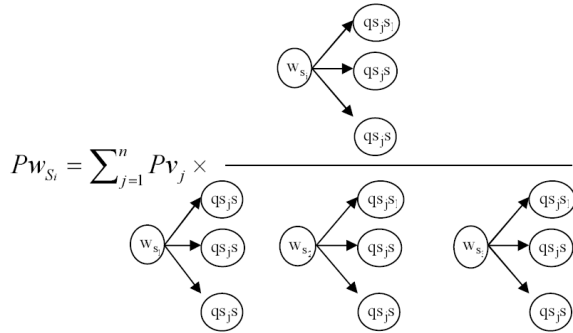


Figure 4: Scoring a sense

4 Experiments

In this section we describe our experiments.

We created a WSM using 90% of SemCor corpus (we did not use tags for training). We evaluated the model with the remaining 10% of SemCor and Senseval-2 (all words nouns only). We chose these corpora to be able to compare with related works such as McCarthy *et al.*

We created a WSM using British National Corpus, and evaluated it with the same data that was used in the previous experiment.

When using a corpus for creating a WSM, the semantic tags of word senses are not considered. These tags refer to specific synsets in WordNet

In these experiments we disambiguated only nouns, because JCN measure is based on WordNet, which does not have populated hierarchies for adjectives or adverbs. Usually verbs are not disambiguated, because they are highly polysemous and the difference between each of their senses is very fine graded.

	Trained on:				
	SemCor	BNC	SemCor	BNC	
	Tested on:				
	Senseval-2		SemCor		
Number of top quasi-synonyms	10	44.22	51.35	64.23	73.07
	20	44.77	52.88	69.44	60.00
	30	45.91	53.33	67.36	65.27
	40	45.76	53.33	66.43	65.16
	50	45.55	53.33	67.8	63.8
	60	48.12	55.36	68.15	63.41
	70	49.84	57.22	69.86	63.84
	100	48.80	56.02	69.86	62.33
	200	49.05	57.57	66.75	61.58
	500	49.10	58.79	65.89	61.08
	1000	44.55	54.27	65.06	61.08
2000	41.05	51.75	62.76	61.08	

Table 1: Precision, training with SemCor and BNC / evaluation with SemCor and Senseval-2

For evaluating, we considered the number of *quasi-synonyms* to choose the right sense. For most of the comparisons, we conducted experiments for the first 10, 20, 30, 40, 50, 60, 70, 100 and 1000 words from the weighted list of *quasi-synonyms*.

In both experiments, general results for 10% of the remaining of Semcor corpus were better than for the Senseval-2 corpus. In the first experiment, the best result using Semcor evaluation was 69.86% precision and in the second one 73.07% precision (See Table 1. Blank cells correspond to experiments not conducted.) These particular results are better than the 64% precision obtained in (McCarthy *et al.* 2004). However, there are many differences, which must be taken into account, between these and McCarthy's experiments: McCarthy used Senseval-2 in the evaluation and the Lin's thesaurus for creating the equivalent of our WSM to obtain a weighted list; also McCarthy's goal was to find the predominant sense whereas our goal was to find the specific sense of an ambiguous word in a context. The results of the second experiment, in which we used the Senseval-2 corpus in our evaluation are better than all the unsupervised methods presented in Senseval-2 (See Table 2).

Rank	Prec.	Recall	System	Sense tagged data?
1	0.69	0.69	SMUaw	Y
2	0.636	0.636	CNTS-Antwerp	Y
3	0.618	0.618	Sinequa-LIA-HMM	Y
4	0.587	0.587	Our Method	N
5	0.575	0.569	UNED - AW-U2	N
6	0.556	0.55	UNED - AW-U	N
7	0.475	0.454	UCLA - gchao2	Y
8	0.474	0.453	UCLA - gchao3	Y
9	0.416	0.451	CL Research - DIMAP	N
10	0.451	0.451	CL Research - DIMAP (R)	N
11	0.5	0.449	UCLA - gchao	Y

Table 2: The Top-10 Systems for Senseval-2

The main goal of this article is to demonstrate how WSD can be improved if we train our method with the same corpus that we use in the evaluation. This hypothesis was confirmed in the first experiment. However, the results obtained in the second experiment did

not confirm our hypothesis entirely. We would expect that training with BNC and evaluating with a fragment of Senseval-2 corpus would be better than evaluating with Semcor. The English Senseval-2 corpus is sampled from BNC and Penn Treebank (comprising components from the Wall Street Journal, Brown, and IBM manuals).

We believe that these surprising results are due to the affinity between Semcor and WordNet, which have been reflected in the measure we have used: JCN. This measure uses the *information content* concept obtained from the SemCor corpus itself in the package WordNet::Similarity. The concept of *information content*, where a value is assigned to the specificity of a concept, was introduced in (Resnik, P. 1995). A concept with a high *information content* is closely related to a particular subject, whereas a concept with a low *information content* is associated to more general subjects. For example, the expression *carving fork* has a high *information content*, while *entity* has a very low *information content*.

5 Conclusions

The method we presented is useful for disambiguating a corpus trained with itself (the first stage consists of training on the corpus itself, the second stage is disambiguation), as shown by the results of training with 90% of SemCor and evaluating with its remaining part. Note that this is not the usual training and test as in supervised learning algorithms, since we are not using sense tags for learning.

Our method obtained better results than all the unsupervised methods presented in Senseval-2. This allows to extend the method proposed in McCarthy *et al.*, which is used for finding the predominant sense of a word in certain corpus, to adaptively use context to find the correct sense of a word using local information.

The method proposed in (McCarthy *et al.* 2004) is used to find the most predominant sense of an ambiguous word considering a weighted list of related terms. In their work, these terms are from the Lin's thesaurus (Lin, D. 1998). This list is always the same for any ambiguous instance of a word, because it does not depend on its context. Our method does not use the Lin's thesaurus. Instead, a specific WSM is created for the corpus to be disambiguated. This way, the list of weighted

terms is not always the same; it depends on the context of the ambiguous word and the corpus wherefrom the resource is created.

The main goal of the method presented in (McCarthy *et al.* 2004) is to obtain the predominant sense for a word, and not the sense expressed in a particular context unless it coincides with the predominant sense; however, the results that they obtained are better than those of any unsupervised method which look for the sense of a word within a context. By substituting the Lin's thesaurus with a syntactic resource built specifically for the corpus to be disambiguated our method improves these results.

Thus, the main difference between the method proposed in (McCarthy *et al.* 2004) and that of ours lies in the list of related terms, which are used by the maximizing algorithm to infer the sense of a word. We can conclude then that the weighted list is an important factor for the disambiguation process in our method.

Another conclusion is about the optimal number of quasi-synonyms that we need in order to disambiguate a word within a specific context. In the first experiment, the results are very irregular; in the second one, the best result was obtained where we used ten quasi-synonyms. The quality of quasi-synonyms seemed to be related with the WSM. In the second experiment the WSM was built with BNC (100 million words) and in the first one with SemCor (1 million word). We believe that strong quasi-synonyms are enough to disambiguate a word with the McCarthy *et al.* algorithm.

The computational cost of our disambiguation algorithm is the same than the one proposed by McCarthy *et al.* The performance of both algorithms depend of WordNet:Similarity package performance and obviously WordNet too.

As a future work, we plan to obtain the *information content* from BNC and repeat the second experiment to see the impact of that concept on the JCN measure and on our method. Also we plan to do testing with wider local contexts. This could be done by considering several levels of the syntactic dependency tree and wider co-occurrence windows, or a combination of both strategies. Finally, we will build a denser WSM using the Google corpus to obtain the strongest possible quasi-synonyms

