Word Sense Disambiguation using a dictionary for sense similarity measure

Bruno Gaume
IRIT – CNRS, UPS & INPT
Toulouse, France
gaume@irit.fr

Nabil Hathout
ERSS – CNRS & UTM
Toulouse, France
hathout@univ-tlse2.fr

Philippe Muller
IRIT – CNRS, UPS & INPT
Toulouse, France
muller@irit.fr

Abstract

This paper presents a disambiguation method in which word senses are determined using a dictionary. We use a semantic proximity measure between words in the dictionary, taking into account the whole topology of the dictionary, seen as a graph on its entries. We have tested the method on the problem of disambiguation of the dictionary entries themselves, with promising results considering we do not use any prior annotated data.

1 Introduction

Various tasks dealing with natural language data have to cope with the numerous different senses possessed by every lexical item: machine translation, information retrieval, information extraction ... This very old issue is far from being solved, and evaluation of methods addressing it is far from obvious (Resnik and Yarowsky, 2000). This problem has been tackled in a number of ways[1]: by looking at contexts of use (with supervised learning or unsupervised sense clustering) or by using lexical resources such as dictionaries or thesauri. The first kind of approach relies on data that are hard to collect (supervised) or very sensitive to the type of corpus (unsupervised). The second kind of approach tries to exploit the lexical knowledge that is represented in dictionaries or thesaurus, with various results from its inception up to now (Lesk, 1986; Banerjee and Pedersen, 2003). In all cases, a distance between words or word senses is used as a way to find the right sense in a given context. Dictionary-based approaches usually rely on a comparison of the set of words used in sense definitions and in the context to disambiguate[2].

This paper presents an algorithm which uses a dictionary as a network of lexical items (cf. sections 2 and 3) to compute a semantic similarity measure between words and word senses. It takes into account the whole topology of the dictionary instead of just the entry of target words. This arguably gives a certain robustness of the results with respect to the dictionary. We have begun testing this approach on word sense disambiguation on definitions of the dictionary itself (section 5), but the method is expected to be more general, although this has not been tested yet. Preliminary results are quite encouraging considering that the method does not require any prior annotated data, while operating on an unconstrained vocabulary.

2 Building the graph of a dictionary

The experiment we describe here has been achieved on a dictionary restricted to nouns and verbs only: we considered dictionary entries classified as nouns and verbs and noun and verb lemmas occurring within those entries. The basic idea is to consider the dictionary as an undirected graph whose nodes are noun entries, and an edge exists between two nodes whenever one of them occur in the definition of the other. More precisely, the graph of the dictionary encodes two types of lexicographical informations: (1) the definitions of the entries sub-senses and (2) the structure of the entries that is the hierarchical organisation of their sub-senses. The graph then includes two types of nodes: w-nodes used for the words that occur

[1] A good introduction is (Ide and Véronis, 1998), or (Manning and Schütze, 1999), chap. 7.

[2] With the exceptions of the methods of (Kozima and Furugori, 1993; Ide and Véronis, 1990), both based on models of activation of lexical relations, but who present no quantifi ed results.

in the definitions and Δ-nodes used for the definitions of the sub-senses of the entries. The graph is created in three phases:

1. For each dictionary entry, there is a Δ-node for the entry as a whole and there is one Δ-node for each of the sub-senses of the entry. Then an edge is added between each Δ-node and the Δ-nodes which represent the sub-senses of the next lower level. In other words, the graph includes a tree of Δ-nodes which encodes the hierarchical structure of each entry.

2. A w-node is created in the graph for each word occurring in a definition and an edge is added between the w-node and the Δ-node of that definition.

3. An edge is added between each w-node and the top Δ-node representing the dictionary entry for that word.

For instance, given the entry for "tooth"[3]:

1. (Anat.) One of the hard, bony appendages which are borne on the jaws, or on other bones in the walls of the mouth or pharynx of most vertebrates, and which usually aid in the prehension and mastication of food.

2. Fig.: Taste; palate.

   These are not dishes for thy dainty tooth. –Dryden.

3. Any projection corresponding to the tooth of an animal, in shape, position, or office; as, the teeth, or cogs, of a cogwheel; a tooth, prong, or tine, of a fork; a tooth, or the teeth, of a rake, a saw, a file, a card.

4. (a) A projecting member resembling a tenon, but fitting into a mortise that is only sunk, not pierced through.

   (b) One of several steps, or offsets, in a tusk. See Tusk.

We would consider one Δ-node for *tooth* as the top-level entry, let us call it Δ_0. Δ_0 is connected with an edge to the Δ-nodes Δ_1, Δ_2 ,Δ_3 and Δ_4 corresponding to the senses 1., 2., 3. and 4.; the latter will have an edge towards the two Δ-nodes $\Delta_{4.1}$ and $\Delta_{4.2}$ for the sub-senses 4.a. and 4.b.; $\Delta_{4.1}$ will have an edge to each w-node built for nouns and verbs occurring in its definition (*member, resemble, tenon, fit, mortise, sink, pierce*). Then the w-node for *tenon* will have an edge to the Δ-node of the top-level entry of *tenon.* We do not directly connect $\Delta_{4.1}$to the Δ-nodes of the top-level entries because these may have both w- and Δ-node daughters.

In the graph, Δ-nodes have tags which indicates their homograph number and their location in the hierarchical structure of the entry. These tags are sequences of integers where the first one gives the homograph number and the next ones indicate the rank of the sense-number at each level. For instance, the previous nodes $\Delta_{4.1}$ is tagged $(0, 4, 1)$.

3 Prox, a distance between graph nodes

We describe here our method (dubbed Prox) to compute a distance between nodes in the kind of graph described in the previous section. It is a stochastic method for the study of so-called hierarchical small-world graphs (Gaume et al., 2002) (see also the next section). The idea is to see a graph as a Markov chain whose states are the graph nodes and whose transitions are its edges, with equal probabilities. Then we send random particles walking through this graph, and their trajectories and the dynamics of their trajectories reveal their structural properties. In short, we assume the average distance a particle has made between two nodes after a given time is an indication of the semantic distance between these nodes. Obviously, nodes located in highly clustered areas will tend to be separated by smaller distance.

Formally, if $G = (V, E)$ is an irreflexive graph with $|V| = n$, we note $[G]$ the $n \times n$ adjacency matrix of G that is such that $[G]_{i,j}$ (the i^{th} row and j^{th} column) is 1 if there is an edge between node i and node j and 0 otherwise.

We note $[\hat{G}]$ the Markovian matrix of G, such that $[\hat{G}]_{r,s} = \frac{[G]_{r,s}}{\sum_{x \in V}([G]_{r,x})}$.

In the case of graphs built from a dictionary as above,$[\hat{G}]_{r,s}$ is 0 if there is no edge between nodes r and s, and $1/D$ otherwise, where D is the number of neighbours of r. This is in-

[3]Source: Webster's Revised Unabridged Dictionary, 1996. The experiment has actually been done on a French dictionary, *Le Robert*.

deed a Markovian transition matrix since the sum of each line is one (the graph considered being connected).

We note $[\hat{G}]^i$ the matrix $[\hat{G}]$ multiplied i times by itself.

Let now PROX(G,i,r,s) be $[\hat{G}]^i_{r,s}$. This is thus the probability that a random particle leaving node r will be in node s after i time steps. This is the measure we will use to determine if a node s is closer to a node r than another node t. Now we still have to find a proper value for i. The next section explains the choice we have made.

4 Dictionaries as hierarchical small-worlds

Recent work in graph theory has revealed a set of features shared by many graphs observed "in the field" These features define the class of "hierarchical small world" networks (henceforth HSW) (Watts and Strogatz, 1998; Newman, 2003). The relevant features of a graph in this respect are the following:

D the density of the network. HSWs typically have a low D, i.e. they have rather few edges compared to their number of vertices.

L the average shortest path between two nodes. It is also low in a HSW.

C the clustering rate. This is a measure of how often neighbours of a vertex are also connected in the graph. In a HSW, this feature is typically high.

I the distribution of incidence degrees (i.e. the number of neighbours) of vertices according to the frequency of nodes (how many nodes are there that have an incidence degree of 1, 2, ... n). In a HSW network, this distribution follows a power law: the probability P(k) that a given node has k neighbours decreases as $k^{-\lambda}$, with $lambda > 0$. It means also that there are very few nodes with a lot of neighbours, and a lot more nodes with very few neighbours.

As a mean of comparison, table 1 shows the differences between randoms graphs (nodes are given, edges are drawn randomly between nodes), regular graphs and HSWs.

The graph of a dictionary belongs to the class of HSW. For instance, on the dictionary we used, D=7, C=0.183, L=3.3. Table 2 gives a few characteristics of the graph of nouns only on the dictionary we used (starred columns indicate values for the maximal self-connected component).

We also think that the hierarchical aspect of dictionaries (reflected in the distribution of incidence degrees) is a consequence of the role of hypernymy associated to the high polysemy of some entries, while the high clustering rate define local domains that are useful for disambiguation. We think these two aspects determine the dynamics of random walks in the graph as explained above, and we assume they are what makes our method interesting for sense disambiguation.

5 Word sense disambiguation using Prox semantic distance

We will now present a method for disambiguating dictionary entries using the semantic distance introduced section (3).

The task can be defined as follows: we consider a word lemma α occurring in the definition of one of the senses of a word β. We want to tag α with the most likely sense it has in this context. Each dictionary entry is coded as a tree of senses in the graph of the dictionary, with a number list associated to each sub-entry. For instance for a given word sense of word W, listed as sub-sense II.3.1 in the dictionary, we would record that sense as a node W(2,3,1) in the graph. In fact, to take homography into account we had to add another level to this, for instance W(1,1,2) is sense 1.2 of the first homograph of word W. In the absence of an homograph, the first number for a word sense will conventionally be 0.

Let G=(V,E) the graph of words built as explained section 2, [G] is the adjacency matrix of G, and $[\hat{G}]$ is the corresponding Markovian matrix . The following algorithm has then been applied:

1. Delete all neighbours of β in G, i.e. make

$$\forall x \in V, [G]_{\alpha,x} = [G]_{x,\alpha} = 0$$

2. Compute the new $[\hat{G}]^i$ where i is taken to be 6

(with equal D)	L	C	I
Random graphs	small L	small C	Poisson Law
HSW	small L	high C	power law
Regular graphs	high L	high C	constant

Table 1: Comparing classes of graphs

	nb nodes	nb edges	nb N*	nb E*	Diam*	C*	L*
Nouns	53770	392161	51511	392142	7	0.1829	3.3249
Nouns and sub-senses	140080	399969	140026	399941	11	0.0081	5.21

Table 2: Dictionary used

3. Let L be the line β in the result. $\forall k, L[k] = [\hat{G}]^i_{\beta,k}$

4. Let $E = \{x_1, x_2, ..., x_n\}$ be the nodes corresponding to all the sub-senses induced by the definition of α.
   Then take $x_k = argmax_{x \in E}(L[x])$

Then x_k is the sub-sense with the best rank according to the Prox distance.

The following steps needs a little explanation:

1 This neighbours are deleted because otherwise there is a bias towards the sub-senses of α, which then form a sort of "artificial" cluster with respect to the given task. This is done to allow the random walk to really go into the larger network of senses.

2 Choosing a good value for the length of the random walk through the graph is not a simple matter. If it is too small, only local relations appear (near synonyms, etc) which might not appear in contexts to disambiguate (this is the main problem of Lesk's method); if it is too large, the distances between words will converge to a constant value. So it has to be related in some way to the average length between two senses in the graph. A reasonable assumption is therefore to stay close to this average length. Hence we took $i = 6$ since the average length is 5.21 (in the graph with a node for every sub-sense, the graph with a node for each entry having L=3.3)

6 Evaluating the results

The following methodology has been followed to evaluate the process.

We have randomly taken about a hundred of sub-entries in the chosen dictionary (out of about 140,000 sub-entries), and have hand-tagged all nouns and verbs in the entries with their sub-senses (and homography number), except those with only one sense, for a total of about 350 words to disambiguate. For all pair of (context,target), the algorithm gives a ranked list of all the sub-senses of the target. Although we have used both nouns and verbs to build the graph of senses, we have tested disambiguation first on *nouns only*, for a total of 237 nouns. We have looked how the algorithm behaves when we used both nouns and verbs in the graph of senses.

To compare the human annotation to the automated one, we have applied the following measures, where $(h_1, h_2, , ...)$ is the human tag, and $(s_1, s_2, ..)$ is the top-ranked system output for a context i defined as the entry and the target word to disambiguate:

1. if $h_1 = 0$ then do nothing else the homograph score is 1 if $h_1 = s_1$, 0 otherwise;

2. in all cases, coarse polysemy count = 1 if $h_2 = s_2$, 0 otherwise;

3. in all cases, fine polysemy count = 1 if $\forall i$ $h_i = s_i$

Thus, the coarse polysemy score computes how many times the algorithm gives a sub-sense that has the same "main" sense as the human tag (the main-sense corresponds to the first level in the hierarchy of senses as defined above). The fine polysemy score gives the number of times the algorithm gives exactly the same sense as the human.

To give an idea of the difficult of the task, we have computed the average number of main

	entry	target	system output	human tag
correct	bal#n._m.*0_3	lieu#n.	**1_1**_3	**1_1**_1
correct	van#n._m.*2_0_0_0_0	voiture#n.	**0_2**	**0_2**_3
error	phonétisme#n._m.*0	moyen#n.	**1**_1_1	**2**_1
error	créativité#n._f.*0	pouvoir#n.	**2_3**	**2_1**
error	acmé#n._m._ou_f.*0_1	phase#n.	**0_1**	**0_4**

Table 3: Detailed, main-sense evaluation of a couple of examples.

sub-senses and the number of all senses, for each target word. This corresponds to a random algorithm, choosing between all senses of a given word. The expected value of this baseline is thus:

- homograph score=$\sum_x$ 1/(number of homographs of x)
- coarse polysemy = $\sum_x$ 1/(number of main sub-senses of x)
- fine polysemy = $\sum_x$ 1/(number of all sub-senses of x)

A second baseline consists in answering always the first sense of the target word, since this is often (but not always) the most common usage of the word. We did not do this for homographs since the order in which they are given in the dictionary does not seem to reflect their importance.

Table 4 sums up the results.

7 Discussion

The result for homographs is very good but not very significant given the low number of occurrences; this all the more true as we used a part-of-speech tagger to disambiguate homographs with different part-of-speech beforehand (these have been left out of the computation of the score).

The scores we get are rather good for coarse polysemy, given the simplicity of the method. As a means of comparison, (Patwardhan et al., 2003) applies various measures of semantic proximity (due to a number of authors), using the WordNet hierarchy, to the task of word sense disambiguation of a few selected words with results ranging from 0.2 to 0.4 with respect to sense definition given in WordNet (the average of senses for each entry giving a random score of about 0.2).

Our method already gives similar results on the fine polysemy task (which has an even harder random baseline) when using both nouns and verbs as nodes, and does not focus on selected targets.

A method not evaluated by (Patwardhan et al., 2003) and using another semantic relatedness measure ("conceptual density") is (Agirre and Rigau, 1996). It is also based on a distance within the WordNet hierarchy. They used a variable context size for the task and present results only for the best size (thus being a not fully unsupervised method). Their random baseline is around 30%, and their precision is 43% for 80% attempted disambiguations.

Another study of disambiguation using a semantic similarity derived from WordNet is (Abney and Light, 1999); it sees the task as a Hidden Markov Model, whose parameters are estimated from corpus data, so this is a mixed model more than a purely dictionary-based model. With a baseline of 0.285, they reach a score of 0.423. Again, the method we used is much simpler, for comparable or better results.

Besides, by using all connections simultaneously between words in the context to disambiguate and the rest of the lexicon, this method avoids the combinatorial explosion of methods purely based on a similarity measure, where every potential sense of every meaningful word in the context must be considered (unless every word sense of words other than the target is known beforehand, which is not a very realistic assumption), so that only local optimization can be achieved. In our case disambiguating a lot of different words appearing in the same context may result in poorer results than with only a few words, but it will not take longer. The only downside is heavy memory usage, as with any dictionary-based method.

We have made the evaluation on dictionary entries because they are already part of the net-

	random	first sense	algorithm(n+v)
homographs	0.49	-	0.875 (14/16)
coarse polysemy	0.35	0.493	0.574 (136/237)
fine polysemy	0.18	0.40	0.346 (82/237)

Table 4: Results

work of senses, to avoid raising other issues too early. Thus, we are not exactly in the context of disambiguating free text. It could then be argued that our task is simpler than standard disambiguation, because dictionary definitions might just be written in a more constrained and precise language. That is why we give the score when taking always the first sense for each entry, as an approximation of the most common sense (since the dictionary does not have frequency information). We can see that this score is about 50% only for the coarse polysemy, and 40% for the fine polysemy, compared to a typical 70-80% in usual disambiguation test sets, for similar sense dispersion (given by the random baseline); in (Abney and Light, 1999), the first-sense baseline gives 82%. So we could in fact argue that disambiguating dictionary entries seems harder. This fact remains however to be confirmed with the actual most frequent senses. Let us point out again that our algorithm does not make use of the number of senses in definitions.

Among the potential sources of improvement for the future, or sources of errors in the past, we can list at least the following:

- overlapping of some definitions for subsenses of an entry. Some entries of the dictionary we used have sub-senses that are very hard to distinguish. In order to measure the impact of this, we should have multiple annotations of the same data and measure inter-annotator agreement, something that has not been done yet.

- part of speech tagging generates a few errors when confusing adjectives and nouns or adjectives and verbs having the same lemma; this should be compensated when we enrich the graph with entries for adjectives.

- some time should be spent studying the precise influence of the length of the random walk considered; we have chosen a value *a priori* to take into account the average length of a path in the graph, but once we have more hand-tagged data we should be able to have a better idea of the best suited value for that parameter.

8 Conclusion

We have presented here an algorithm giving a measure of lexical similarity, built from information found in a dictionary. This has been used to disambiguate dictionary entries, with a method that needs no other source of information (except part-of-speech tagging), no annotated data. The coverage of the method depends only on the lexical coverage of the dictionary used. It seems to give promising results on disambiguating nouns, using only nouns or nouns and verbs. We intend to try the method after enriching the network of senses with adjectives and/or adverbs. We also intend, of course, to try the method on disambiguating verbs and adjectives.

Moreover, the method can be rather straightforwardly extended to any type of disambiguation by considering a context with a target word as a node added in the graph of senses (a kind of virtual definition). We have not tested this idea yet. Since our method gives a ranked list of sense candidates, we also consider using finer performance measures, taking into account confidence degrees, as proposed in (Resnik and Yarowsky, 2000).

References

Steven Abney and Marc Light. 1999. Hiding a semantic hierarchy in a markov model. In *ACL'99 Workshop Unsupervised Learning in Natural Language Processing*, University of Maryland.

E. Agirre and G. Rigau. 1996. Word sense disambiguation using conceptual density. In *Proceedings of COLING'96*, pages 16–22, Copenhagen (Denmark).

S. Banerjee and T. Pedersen. 2003. Extended gloss overlaps as a measure of semantic re-

latedness. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence (IJCAI-03)*, Acapulco, Mexico.

B. Gaume, K. Duvignau, O. Gasquet, and M.-D. Gineste. 2002. Forms of meaning, meaning of forms. *Journal of Experimental and Theoretical Artificial Intelligence*, 14(1):61–74.

N. Ide and J. Véronis. 1998. Introduction to the special issue on word sense disambiguation: The state of the art. *Computational Linguistics*, 24(1).

N. Ide and J. Véronis. 1990. Word sense disambiguation with very large neural networks extracted from machine readable dictionaries. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING'90)*, volume 2, pages 389–394.

H. Kozima and T. Furugori. 1993. Similarity between words computed by spreading activation on an english dictionary. In *Proceedings of the conference of the European chapter of the ACL*, pages 232–239.

M. Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine code from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26, Toronto, Canada.

C. Manning and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

M. E. J. Newman. 2003. The structure and function of complex networks. *SIAM Review*, 45:167–256.

S. Patwardhan, S. Banerjee, and T. Pedersen. 2003. Using measures of semantic relatedness for word sense disambiguation. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics (CICLING-03)*.

P. Resnik and D. Yarowsky. 2000. Distinguishing systems and distinguishing senses: New evaluation methods for word sense disambiguation. *Natural Language Engineering*, 5(2):113–133.

D.J. Watts and S.H Strogatz. 1998. Collective dynamics of 'small-world' networks. *Nature*, (393):440–442.