RACAI: Meaning Affinity Models

Radu Ion
Institute for Artificial Intelligence
13, "13 Septembrie",
050711, Bucharest 5,
Romania
radu@racai.ro

Dan Tufiş
Institute for Artificial Intelligence
13, "13 Septembrie",
050711, Bucharest 5,
Romania
tufis@racai.ro

Abstract

This article introduces an unsupervised word sense disambiguation algorithm that is inspired by the lexical attraction models of Yuret (1998). It is based on the assumption that the meanings of the words that form a sentence can be best assigned by constructing an interpretation of the whole sentence. This interpretation is facilitated by a dependency-like context specification of a content word within the sentence. Thus, finding the context words of a target word is a matter of finding a pseudo-syntactic dependency analysis of the sentence, called a linkage.

1 Introduction

Word Sense Disambiguation (WSD) is a difficult Natural Language Processing task which requires that for every content word (noun, adjective, verb or adverb) the appropriate meaning is automatically selected from the available sense inventory[1]. Traditionally, the WSD algorithms are divided into two rough classes: supervised and unsupervised. The supervised paradigm relies on sense annotated corpora, with the assumption that neighbouring disambiguate words provide a strongly discriminating and generalizable context representation for the meaning of a target word. Obviously, this approach suffers from the *knowledge acquisition bottleneck* in that there will never be enough training data to ensure a scalable result of such algorithms. The unsupervised alternative to WSD tries to alleviate the burden of manually sense tagging the corpora, by employing algorithms that use different knowledge sources to determine the correct meaning in context. In fact, the "knowledge source usage" is another way to distinguish among the WSD methods. Such methods call upon further processing of the text to be disambiguated such as parsing and/or use handcrafted, semantically rich sense inventories such as WordNet (Fellbaum, 1998). WSD methods in this category range from the very simple ranking based on counting the number of words occurring in both the target word's context and its sense definitions in a reference dictionary (Lesk, 1986) to the more elaborated approaches using the semantic lexicon's taxonomies, (shallow) parsing, collocation discovery etc. (Stevenson and Wilks, 2001).

One of the central issues of any WSD implementation is given by the *context representation*. The standard principle that is applied when trying to disambiguate the meaning of a word is that the same word in similar contexts should have the same meaning. By and large, the context of a target word is materialized by a collection of features among which are: the collocates of the target word, the part-of-speech (POS) of the target word, $\pm k$ words surrounding the target word and/or their POSes and so on. More often than not, the contexts similarity is estimated by the distance in the feature vector space. Lin (1997) defines the local context of a target word by the collection of syntactic dependencies in which the word takes part. According to this notion of con-

[1] In principle, one can select meanings for any part of speech that is represented into the semantic lexicon (prepositions for instance) but the content words disambiguation is the de facto standard.

text, Lin assumes that two different words are likely to have similar meanings if they occur in identical local contexts.

What we will attempt here is to combine the two views of context similarity/identity versus meaning similarity/identity by using a dependency-like representation of the context as a lexical attraction model. More specifically, we will not consider any feature of the context and will try to maximize a meaning attraction function over all linked words of a sentence. In section 2 we will describe **SynWSD**, an unsupervised, knowledge-based WSD algorithm and in sections 3 and 4 we will present the application of SynWSD to two of SEMEVAL-2007 "all words" tasks: English Coarse-Grained and English Fine-Grained. Finally, with section 5 we will conclude the article.

2 SynWSD

The syntactic context representation is not new in the realm of WSD algorithms. For instance, Lin (1997) used the dependency relations of the target word to specify its context and Stetina (1998) extracted head-modifier relations to obtain the context pairs for each word of interest from a constituents tree. The syntactic representation of the context of a target word has one main advantage over the collection of features method: the target word is related only with the relevant word(s) in its window and not with all the words and thus, many noisy cooccurrences are eliminated. Mel'čuk (1988) further strengthens the intuition of a syntactic context representation with his Meaning Text Model in which there is a deterministic translation from the surface syntactic dependency realization of the sentence to its deep syntactic one and therefore to the semantic representation.

To use a syntactic analysis as a context representation, one needs a parser which will supply the WSD algorithm with the required analysis. Because we have intended to develop a language independent WSD algorithm and because there is no available, reliable dependency parser for Romanian, we have backed off to a simpler, easier to obtain dependency-like representation of a sentence: a slightly modified version of the lexical attraction models of (Yuret, 1998).

2.1 LexPar

Lexical attraction is viewed as the likelihood of a syntactic dependency relation between two words of a sentence and is measured by the pointwise mutual information between them. Yuret (1998) shows that the search for the lowest entropy lexical attraction model leads to the unsupervised discovery of undirected dependency relations or links.

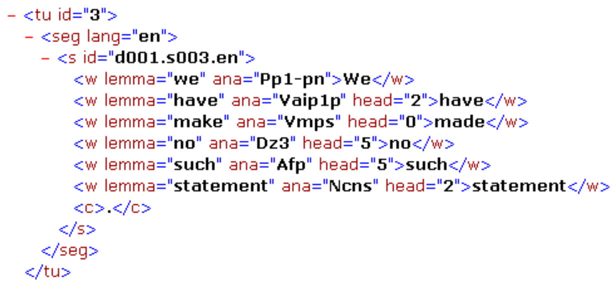
LexPar (Ion and Barbu Mititelu, 2006) is a link analyzer (a linker) which generates a connected, undirected, acyclic and planar graph of an input sentence in which the nodes are the words of the sentence and the edges are the highest lexical attracted dependency-like relations. This program is similar to the suboptimal one presented in (Yuret, 1998) with the following main differences:

- the policy of checking pairs of words to be related is based on the assumption that most of the syntactic relations[2] are formed between adjacent words and then between adjacent groups of linked words;
- it operates on POS-tagged and lemmatized corpora and attempts to improve parameter estimation by using both lemmas and POS tags. The score of a link is defined as the weighted sum of the pointwise mutual information of the lemmas and of the POS tags, thus coping even with the unknown lemmas;
- it uses a rule filter that will deny the formation of certain links based on the POSes of the candidate words. For instance, neither the relation between a determiner and an adverb nor the relation between a singular determiner and a plural noun should be permitted;

In Figure 1 we have an example of a XML encoded, LexPar processed sentence. The `head` attribute of the `w` tag specifies the position of the head word of the tagged word. Because LexPar considers non-directed dependency relations, for the purposes of XML encoding[3], the first word of every sentence

[2]At least for our languages of interest, namely English and Romanian.

[3]The encoding of the morpho-syntactic descriptors (MSD) is MULTEXT-East compliant (`http://nl.ijs.si/ME/V3/msd/00README.txt`).



```
- <tu id="3">
  - <seg lang="en">
    - <s id="d001.s003.en">
        <w lemma="we" ana="Pp1-pn">We</w>
        <w lemma="have" ana="Vaip1p" head="2">have</w>
        <w lemma="make" ana="Vmps" head="0">made</w>
        <w lemma="no" ana="Dz3" head="5">no</w>
        <w lemma="such" ana="Afp" head="5">such</w>
        <w lemma="statement" ana="Ncns" head="2">statement</w>
        <c>.</c>
      </s>
    </seg>
  </tu>
```

Figure 1: The XML representation of a LexPar processed sentence.

(position 0) is always the root of the syntactic dependency tree, its dependents are its children nodes, and so on while we recursively build the tree from the LexPar result.

We have chosen not to give a detailed presentation of LexPar here (the reader is directed towards (Yuret, 1998; Ion and Barbu Mititelu, 2006)) and instead, to briefly explain how the linkage in Figure 1 was obtained. The processor begins by inspecting a list G of groups of linked words which initially contains the positions of each of the words in the sentence:

$$G_0 = \{(0),(1),(2),(3),(4),(5)\}$$

The linking policy is trying to link words in the groups (0) and (1) or (1) and (2). The syntactic rule filter says that auxiliary verbs (`Va`) can only be linked with main verbs (`Vm`) and so one link is formed and the list of groups becomes:

$$G_1 = \{(0),(\langle 1,2\rangle),(3),(4),(5)\}$$

Next, the processor must decide linking the groups $(\langle 1,2\rangle)$ and (3) or (3) and (4) but the syntactic rule filter is denying any link from positions 1 or 2 to 3 (no links from any kind of verb `V` to any kind of a determiner `D`) or from 3 to 4 (no link from a negative determiner `Dz3` to a qualificative adjective `Af`). Continuing this way, the progress of G list is as follows:

$$\begin{aligned} G_1 &= \{(0),(\langle 1,2\rangle),(3),(\langle 4,5\rangle)\} \\ G_2 &= \{(\langle 0,2\rangle,\langle 1,2\rangle),(\langle 3,5\rangle,\langle 4,5\rangle)\} \\ G_3 &= \{(\langle 0,2\rangle,\langle 1,2\rangle,\langle 2,5\rangle,\langle 3,5\rangle,\langle 4,5\rangle)\} \end{aligned}$$

So in 3 steps G_3 contains a single group of linked words namely the linkage of the sentence.

2.2 Meaning Affinity Models

If the lexical attraction models are geared towards the discovery of the most probable syntactic relations of a sentence, we can naturally generalize this idea to construct a class of models that will find a combination of meanings that maximizes a certain meaning attraction function over a linkage of a sentence. We call this class of models the *meaning affinity models*.

Optimizing meaning affinity over a syntactic representation of a sentence has been tried in (Stetina et al., 1998; Horbovanu, 2002). SynWSD (Ion, 2007) is an implementation with two phases of the meaning affinity concept: **training** which takes as input a corpus with LexPar linked sentences (of the type shown in Figure 1) and outputs a table M of meaning co-occurrence frequencies and **disambiguation** of a LexPar linked sentence S, based on the counts in table M from the previous phase.

Before continuing with the descriptions of these phases, we will introduce the notations that we will use throughout this section:

- A n-word sentence is represented by a vector S of n elements, each of them containing a triple $\langle \text{wordform}, \text{lemma}, \text{POS}\rangle$. For instance, the first element from S in Figure 1 is $S[0] = \langle \text{We}, \text{we}, \texttt{Pp1-pn}\rangle$;
- L is the LexPar linkage of S, and is also a vector containing pairs of positions $\langle i, j\rangle$ in S that are related, where $0 \leq i < j < n$;
- $\texttt{lem}(S, i)$ and $\texttt{pos}(S, i)$ are two functions that give the lemma and the POS of the position i in S, $0 \leq i < n$.

The **training phase** is responsible for collecting meaning co-occurrence counts. It simply iterates over each sentence S of the training corpus and for every link $L[k]$ of the form $\langle a, b\rangle$ from its linkage, does the following (K stores the total number of recorded meaning pairs):

1. extracts the sets of meanings I_a and I_b corresponding to the lemma $\texttt{lem}(S, a)$ with the POS $\texttt{pos}(S, a)$ and to the lemma $\texttt{lem}(S, b)$ with the POS $\texttt{pos}(S, b)$ from the sense inventory[4];

[4]If the lemma does not appear in the sense inventory or its

2. increases by 1 the M table frequencies for every pair of the cartesian product $I_a \times I_b$. For every meaning $m \in I_a$, the frequency of the special pair $\langle m, * \rangle$ is increased with $|I_b|$. Similarly, the pair $\langle *, m \rangle$ frequency is also increased with $|I_a|$ for $m \in I_b$);

3. $K \leftarrow K + |I_a \times I_b|$.

We have used the Princeton WordNet (Fellbaum, 1998), version 2.0 (PWN20) as our sense inventory and the mappings from its synsets to the SUMO ontology concepts (Niles and Pease, 2003) and to the IRST domains (Magnini and Cavaglia, 2000). Thus we have tree different sense inventories each with a different granularity. For instance, the noun **homeless** has 2 senses in PWN20, its first sense ("*someone with no housing*") being mapped onto the more general `Human` SUMO concept and onto the `person` IRST domain. The second sense of the same noun is "*people who are homeless*" which corresponds to the same SUMO concept and to a different IRST domain (`factotum`).

In order to reduce the number of recorded pairs in the case of PWN20 meanings (the finest granularity available) and to obtain reliable counts, we have modified the step 1 of the training phase in the following manner:

- if we are dealing with nouns or verbs, for every meaning m_i of the lemma, extract the uppermost hypernym meaning which does not subsume any other meaning of the same lemma;
- if we are dealing with adjectives, for every meaning m_i of the lemma, extract the meaning of the head adjective if m_i is part of a cluster;
- if we are dealing with adverbs, for every meaning m_i of the lemma, return m_i (no generalization is made available by the sense inventory in this case).

This generalization procedure will be reversed at the time of disambiguation as will be explained shortly.

POS does not give a noun, verb, adjective or adverb, the lemma itself is returned as the sole meaning because in the disambiguation phase we need a meaning for every word of the sentence, be it content word or otherwise.

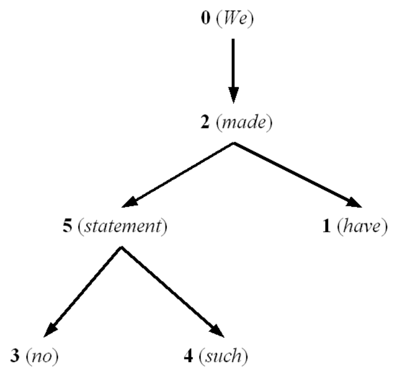


Figure 2: The tree representation of the sentence in Figure 1.

The **disambiguation phase** takes care of finding the best interpretation of a linked sentence based on the frequency table M. For a test sentence S, with the linkage L, the procedure goes as follows:

1. produce a proper tree T of positions from L by taking position 0 as the root of the tree. Then, for every link that contains 0 make the other position in the link a child of 0 and then, in a recursive manner, apply the same process for all children of 0. For instance, the tree for Figure 1 if depicted in Figure 2;

2. construct a vector P of sentence positions visited during a depth-first traversal of the T tree. The vector of sentence positions for Figure 2 is

$$P = (0, 2, 5, 3, 5, 4, 5, 2, 1, 2, 0)$$

3. construct a meaning vector V of the same length as P. $V[i]$ contains the list of meanings of the lemma $\texttt{lem}(S, P[i])$ with the POS $\texttt{pos}(S, P[i])$. If the sense inventory is PWN20, every meaning from the list is generalized as described above;

4. finally, apply the Viterbi algorithm ((Viterbi, 1967)) on the V vector and extract the path (sequence of meanings) which maximizes meaning affinity.

Each state transition is scored according to a meaning affinity function. In our experiments we have considered three meaning affinity functions. If K is the total number of meaning pairs and if m_1

and m_2 are two meanings from adjacent V positions for which $f(m_1, m_2)$ is the pair frequency extracted from M, the functions are:

1. DICE:

$$\texttt{dice}(m_1, m_2) = \frac{2f(m_1,m_2)+2f(m_2,m_1)}{f(m_1,*)+f(*,m_1)+f(m_2,*)+f(*,m_2)}$$

2. Pointwise mutual information:

$$\texttt{mi}(m_1, m_2) = log\frac{Kf(m_1,m_2)+Kf(m_2,m_1)}{(f(m_1,*)+f(*,m_1))(f(m_2,*)+f(*,m_2))}$$

3. Log-Likelihood, $\texttt{ll}(m_1, m_2)$ which is computed as in (Moore, 2004).

After the Viterbi path (best path) has been calculated, every state (meaning) from $V[i]$ ($0 \leq i < |V|$) along this path is added to a final D vector. When the PWN20 sense inventory is used, the reverse of the generalization procedure is applied to each meaning recorded in D, thus coming back to the meanings of the words of S. Please note that an entry in D may contain more than one meaning especially in the case of PWN20 meanings for which there was not enough training data.

3 SEMEVAL-2007 Task #7: Coarse-grained English All-Words

LexPar and SynWSD were trained on an 1 million words corpus comprising the George Orwell's 1984 novel and the SemCor corpus (Miller et al., 1993). Both texts have been POS-tagged (with MULTEXT-East compliant POS tags) and lemmatized and the result was carefully checked by human judges to ensure a correct annotation.

SynWSD was run with all the meaning attraction functions (`dice`, `mi` and `ll`) for all the sense inventories (PWN20, SUMO categories and IRST domains) and a combined result was submitted to the task organizers. The combined result was prepared in the following way:

1. for each sense inventory and for each token identifier, get the union of the meanings for each run (`dice`, `mi` and `ll`);
2. for each token identifier with its three union sets of PWN20 meanings, SUMO categories and IRST domains:
   (a) for each PWN20 meaning m_i in the union, if there is a SUMO category that maps onto it, increase m_i's weight by 1;
   (b) for each PWN20 meaning m_i in the union, if there is a IRST domain that maps onto it, increase m_i's weight by 1;
   (c) from the set of weighted PWN20 meanings, select the subset C that best overlaps with a cluster. That is, the intersection between the subset and the cluster has a maximal number of meanings for which the sum of weights is also the greatest;
   (d) output the lowest numbered meaning in C.

With this combination, the official F-measure of SynWSD is 0.65712 which places it into the 11^{th} position out of 16 competing systems[5].

Another possible combination is that of the intersection which is obtained with the exact same steps as above, replacing the union operation with the intersection. When the PWN20 meanings set is void, we can make use of the most frequent sense (MFS) backoff strategy thus selecting the MFS of the current test word from PWN20. Working with the official key file and scoring software, the intersection combination with MFS backoff gives an F-measure of 0.78713 corresponding to the 6^{th} best result. The same combination method but without MFS backoff achieves a precision of 0.80559 but at the cost of a very low F-measure (0.41492).

4 SEMEVAL-2007 Task #17: English All-Words

For this task, LexPar and SynWSD were further trained on a 12 million POS tagged and lemmatized balanced corpus[6]. The run that was submitted was the intersection combination with the MFS backoff strategy which obtained an F-measure of 0.527. This score puts our algorithm on the 8^{th} position out of 14 competing systems. For the union combinator

[5]Precision = Recall = F-measure. In what follows, mentioning only the F-measure means that this equality holds.

[6]A random subset of the BNC (`http://www.natcorp.ox.ac.uk/`).

(the MFS backoff strategy is not applicable), the F-measure decreases to 0.445 (10^{th} place). Finally, if we train SynWSD only on corpora from task#7, the union combinator leads to an F-measure of 0.344.

5 Conclusions

SynWSD is a knowledge-based, unsupervised WSD algorithm that uses a dependency-like analysis of a sentence as a uniform context representation. It is a language independent algorithm that doesn't require any feature selection.

Our system can be improved in several ways. First, one can modify the generalization procedure in the case of PWN20 meanings in the sense of selecting a fixed set of top level hypernyms. The size of this set will directly affect the quality of meaning co-occurrence frequencies. Second, one may study the effect of a proper dependency parsing on the results of the disambiguation process including here making use of the syntactic relations names and orientation.

Even if SynWSD rankings are not the best available, we believe that the unsupervised approach to the WSD problem combined with different knowledge sources represents the future of these systems even if, at least during the last semantic evaluation exercise SENSEVAL-3, the supervised systems achieved top rankings.

References

Christiane Fellbaum, editor. 1998. *WordNet. An Electronic Lexical Database*. MIT Press, May.

Vladimir Horbovanu. 2002. Word Sense Disambiguation using WordNet. "Alexandru Ioan Cuza" University, Faculty of Computer Science, Iaşi, Romania. In Romanian.

Radu Ion and Verginica Barbu Mititelu. 2006. Constrained lexical attraction models. In *Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference*, pages 297–302, Menlo Park, Calif., USA. AAAI Press.

Radu Ion. 2007. *Word Sense Disambiguation methods applied to English and Romanian*. Ph.D. thesis, Research Institute for Artificial Intelligence (RACAI), Romanian Academy, January. In Romanian, to be defended.

Michael Lesk. 1986. Automatic sense disambiguation : How to tell a pine cone from an ice cream cone. In *Proceedings of the 1986 SIGDOC Conference, Association for Computing Machinery*, pages 24–26, New York.

Dekang Lin. 1997. Using syntactic dependency as local context to resolve word sense ambiguity. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 64–71, Madrid, Spain, July.

Bernardo Magnini and Gabriela Cavaglia. 2000. Integrating Subject Field Codes into WordNet. In Gavrilidou M., Crayannis G., Markantonatu S., Piperidis S., and Stainhaouer G., editors, *Proceedings of LREC-2000, Second International Conference on Language Resources and Evaluation*, pages 1413–1418, Athens, Greece, June.

Igor Mel'čuk. 1988. *Dependency Syntax: theory and practice*. State University of New York Press, Albany, NY.

George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *Proceedings of the 3rd DARPA Workshop on Human Language Technology*, pages 303–308, Plainsboro, New Jersey.

Robert C. Moore. 2004. On Log-Likelihood Ratios and the Significance of Rare Events. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 333–340, Barcelona, Spain.

Ian Niles and Adam Pease. 2003. Linking Lexicons and Ontologies: Mapping WordNet to the Suggested Upper Merged Ontology. In *Proceedings of the 2003 International Conference on Information and Knowledge Engineering (IKE 03)*, Las Vegas, Nevada, June.

Jiri Stetina, Sadao Kurohashi, and Makoto Nagao. 1998. General word sense disambiguation method based on a full sentential context. In *Proceedings of the Coling-ACL'98 Workshop "Usage of WordNet in Natural Language Processing Systems"*, pages 1–8, Montreal.

Mark Stevenson and Yorick Wilks. 2001. The interaction of knowledge sources in word sense disambiguation. *Computational Linguistics*, 27(3):321–349.

Andrew J. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT(13):260–269, April.

Deniz Yuret. 1998. *Discovery of linguistic relations using lexical attraction*. Ph.D. thesis, Department of Computer Science and Electrical Engineering, MIT, May.