A Client/Server Architecture for Word Sense Disambiguation

Caroline Brun
Xerox Research Centre Europe
6, chemin de Maupertuis
38240 Meylan France
Caroline.Brun@xrce.xerox.com

Abstract

This paper presents a robust client/server implementation of a word sense disambiguator for English. This system associates a word with its meaning in a given context using dictionaries as tagged corpora in order to extract semantic disambiguation rules. Semantic rules are used as input of a semantic application program which encodes a linguistic strategy in order to select the best disambiguation rule for the word to be disambiguated. The semantic disambiguation rule application program is part of the client/server architecture enabling the processing of large corpora.

1 Introduction

This paper describes the implementation of an on-line lexical semantic disambiguation system for English within a client/server linguistic application. This system allows to select the meaning of a word given its context of appearance in a text segment, and addresses the general problem of Word Sense Disambiguation (WSD), (Ide et al90), (Gale et al. 92), (Gale et al. 93), (Leacock et al.93), (Yarowsky 95), (Ng et al. 96), (Resnik et al. 97), (Véronis et al. 98) and (Wilks et al. 98).

The basic idea of the semantic disambiguation system described here is to use a dictionary, in our case, the Oxford-Hachette bilingual dictionary (OHFD), (Oxford 94), a bilingual English/French French/English dictionary designed initially for humans but stored in SGML format, in order to extract a semantic disambiguation rule database. The dictionary is in effect used as a semantically tagged corpus.

Once the semantic disambiguation database is available, it becomes, as well as a dictionary and an ontology, a resource used by the server to perform WSD on new input. A linguistic strategy was implemented in order to select the best matching disambiguation rule in a given context.

This implementation is a follow-up of the Semantic Dictionary Lookup (SDL) already implemented in this client/server system (Aimelet 98) and of the methods proposed in (Dini et al. 98) and (Dini et al. 99). The originality of our implementation lies in the rule selection strategy for application as well as in the use of the client/server characteristics to perform WSD. After a brief presentation of the client/server characteristics, we examine the implementation of the WSD system. Then we describe the results obtained after evaluation of the system, and finally we conclude with the description of its applications and perspectives.

2 Architecture of the system

2.1 XeLDa: a linguistic client/server application

XeLDa addresses the problem of a generic development framework for linguistic-based and linguistics-enriched applications, based on available, as well as future research results. Potential applications include: translation aids over a network, on a desktop or a portable PC, syntax checking, terminology extraction, and authoring tools in general. This system provides developers and researchers with a common development architecture for the open and seamless integration of linguistic services. XeLDa offers different services such as dictionary lookup, tokenization, tagging, shallow parsing, etc.

Dictionary lookup and shallow parsing are extensively used in the semantic rule extraction/application processes described in this paper.

2.2 Dictionary Lookup

The OHFD dictionary is accessible via the XeLDa server, which allows a fast and easy lookup of words. Each entry in the OHFD dictionary (cf. Fig1, entry of *seize* in SGML format) is organized in different levels (Akroyd 92), corresponding to syntactic categories (<S1> ... </S1>, S1=part of speech

distinction[1]), which are themselves divided into semantic categories (<S2> ... </S2>, the senses we are interested in), themselves divided into several translations (<TR> ... </TR>).

```
<SE>
<HW>seize</HW>
<HG><PR><PH>si:z</PH></PR></HG>
 <S1><O1><PS>vtr</PS></O1>
   <S2><O2><LA>lit</LA>
           <IC>take hold of</IC></O2>
       <TR>saisir<CO>person,
                  object</CO></TR>
       <TR><LE>to seize sb around the
              waist</LE>saisir qn par
              la taille</TR>
       <TR><LI>to seize hold of</LI>
       se saisir de<CO>person</CO></TR>
       <TR>s'emparer de<CO>object</CO>
       </TR>
       <TR>sauter sur<CO>idea</CO></TR>
   </S2>
   <S2><O2>
     <LA>fig</LA><IC>grasp</IC></O2>
       <TR>saisir<CO>opportunity,
                  moment</CO></TR>
       <TR>prendre<CO>initiative</CO>
       </TR>
       <TR><LI>to be seized by</LI>
       être pris de<CO>emotion,
                  pain, fit</CO></TR>
   </S2>
   <S2><O2><LA>Mil</LA><LA>Pol</LA>
       <IC>capture</IC></O2>
       <TR>s'emparer de<CO>power,
       territory, hostage, prisoner,
       installation</CO></TR>
       <TR>prendre<CO>control</CO></TR>
   </S2>
   <S2><O2<LA>Jur</LA></O2>
       <TR>saisir<CO>arms, drugs,
                  property</CO></TR>
       <TR>appréhender<CO>person</CO>
       </TR>
   </S2>
 </S1>
 <S1><O1>
     <PS>vi</PS></O1>
     <TR><CO>engine, mechanism</CO>
                  se gripper</TR>
   </S1>
</SE>
```

Fig1: SGML entry of *seize*

---

[1]S1 are a bit more informative than simple part of speech since they distinguish also transitive, intransitive reflexive verbs, past participles, as well as some plural/singular nouns.

Fine-grained SGML tags mark up different kinds of information related to semantic categories (<S2>) and translations, in particular:

- <CO> ... </CO> mark collocates (typical subjects, objects, modifiers,...);
- <LC> ... </LC> mark compound examples associated with the headword ;
- <LE> ... </LE> mark general examples used for illustration of a word or a phrase;
- <LI> ... </LI> mark idiomatic examples;
- <LO> ... </LO> mark examples illustrating an obligatory syntactic structure of an entry;
- <LU> ... </LU> mark examples of usage;
- <LV> ... </LV> mark examples of phrasal verb pattern.

The meta-semantic information encoded into these different SGML tags is used to acquire semantic disambiguation rules from the dictionary and guides the semantic rule application process, as explained later.

2.3 Shallow Parser

The "shallow parsing" technology is based on a cascade of finite state transducers which allows us to extract from a sentence its shallow syntactic structure (chunks) and its functional relationships (Aït et al. 97).

The following example illustrates the kind of analysis provided by the shallow parser:

A revolver and two shotguns were seized at the party.

```
[SC [NP A revolver NP]/SUBJ and
[NP two shotguns NP]/SUBJ :v were
seized SC] [PP at the party PP].

SUBJPASS(revolver,seize)
SUBJPASS(shotgun,seize)
VMODOBJ(seize,at,party)
```

Shallow parser transducers are accessible via the XeLDa server enabling fast and robust execution (Roux98).

The syntactic relations used in the disambiguation system are subject-verb, verb-object and modifier. Subject-verb relations include cases such as passives, reflexive and relative constructions. Modifier

relations includes nominal, prepositional, adjectival, and adverbial phrases as well as relative clauses.

2.4 Rule extractor

To perform semantic tag assignment using the OHFD dictionary, a sense number (S_i) is assigned to each semantic category (<S2>) of each entry. These sense numbers act as semantic tags in the process of disambiguation rule application, because they directly point to a particular meaning of an entry.

In the context of our OHFD-based implementation, sense numbering consists in concatenating the homograph number (which is 0 if there are no homographs of the entry, or 1, 2, 3, ..., for each homograph otherwise), the S1 number, and the S2 number. For example, the entry *seize* is composed of five distinct senses, respectively numbered 0.I.1, 0.I.2, 0.I.3, 0.I.4 (for the transitive verb), 0.II.1 (for the intransitive verb). Such sense numbers allow a deterministic retrieval of the semantic categories of a word.

As in GINGER I (Dini et al. 98) and GINGER II (Dini et al. 99) the acquired rules are of two types: *word level* and/or ambiguity *class level*.

The database is built according to the following strategy: for each sense number S_i of the entry, examples are parsed with the shallow parser, and functional dependencies are extracted from these examples: if a dependency involves the entry lemma (headword), a semantic disambiguation rule is built. It can be paraphrased as:

If the lemma X, which is ambiguous between S_1, S_2, ..., S_n, appears in the dependency DEP(X,Y) or DEP(Y,X) then it can be disambiguated by assigning the sense S_i.

Such rules are *word level* rules, because they match the lexical context.

For each sense number again, collocates are used to build semantic rules. The type of dependency illustrated by a collocate of an entry is SGML-tagged in the OHFD[2], and is directly exploited to build rules in the same way.

Then, for each rule already built, semantic classes from an ontology (in our case, WordNet[3], (Fell-

---

[2]For example, a collocate in a verb entry describes either a SUBJ or an OBJ dependency depending on its SGML tag

[3]Since WordNet classes are relatively poor for adjectives and adverbs, additional information about adjectival and adverbial classes is extracted from a general thesaurus, the Roget.

baum 98)) are used to generalize the scope of the rules: the non-headword argument of functional dependencies is replaced in the rule by its semantic classes. The resulting rule can be paraphrased as:

If the lemma X, which is ambiguous between S_1, S_2, ..., S_n, appears in the dependency DEP(X, ambiguity_class(Y)) or DEP(ambiguity_class(Y), X) then it can be disambiguated by assigning the sense S_i.

Such rules are *class level* rules, because they match the semantic context rather than lexical items. In both cases, the type of the rule (<LC>, <LE>, <LI>, <LO>, <LU>, <LV>, <CO>) is kept and encoded into the rules.

For example, from the last semantic category of *seize*, 0.I.1, the system built the following word level rules:

SUBJ(engine,seize) ⇒ 0.I.1 <CO>;
SUBJ(mechanism,seize) ⇒ 0.I.1 <CO>;

Since *engine* belongs to the classes number 6 (noun.artifact) and 19 (noun.phenomenon), whereas *mechanism* belongs to the classes number 6, 4 (noun.act), 17 (noun.object), and 22 (noun.process), corresponding class level rules are:

SUBJ(6/19,seize) ⇒ 0.I.1 <CO>;
SUBJ(4/6/17/22,seize) ⇒ 0.I.1 <CO>;

All dictionary entries are processed, which allow to automatically build a semantic disambiguation rule database available to be used by the semantic application program to disambiguate unseen texts.

2.5 Rule application program

The **rule application program** matches rules of the semantic database against new unseen input text using a preference strategy in order to disambiguate words on the fly. In cases where the system is not able to find any matching rules, it gives as fall back result the first meaning corresponding to the syntactic part of speech of the word in the sentence. Since the OHFD has been built using corpora frequencies, the most frequent senses of a word appear first in the entry. Therefore, even if there are no matching rules, the system gives as result the most probable meaning of the word to disambiguate.

The linguistic strategy used in the application program is shown on several examples.

2.5.1 Simple rule matching

Suppose one wants to disambiguate the word *seize* in the sentence:

Only after oranges had been served did Jed seize the initiative, a scrummage pick-up effort by Ronnie Kirkpatrick cancelling out Moore's score.

The rule application program first extracts the functional dependencies by means of the shallow parser. The word to be disambiguated has to be member of one or more dependencies, in this case:

DOBJ(seize,initiative)

The next step tries to match these dependencies with one or more rules in the semantic disambiguation database.

If one and only one rule matches the lexical context of the dependencies directly, the system uses it to disambiguate the word, i.e. to assign the sense number S_i[4] to it; otherwise, if several rules match directly at word level, the selection process uses the meta-semantic information encoded in SGML tags within the dictionary (and kept in the rules on purpose) with the following preference strategy: rule built from collocate (<CO>), from compounds examples (<LC>), from idiomatic examples (<LI>), from structure examples (<LO>), from phrasal verb pattern examples (<LV>), from usage examples (<LU>), and finally from general examples (<LE>). As far as implementation is concerned, rules are weighted from 1 to 7 according to their types. This strategy relies on the linguistic choices lexicographers made to build the dictionary and takes into account the accuracy of the linguistic type of the examples: it ranges from collocates, which encode very typical arguments of predicates, to very general examples, as such the resulting rules are linguistically-based.

In these particular example, only one lexical rule matches the dependency extracted:

seize: DOBJ(seize,initiative) $\Rightarrow$ 0.I.2 <CO>

meaning that the sense number affected to *seize* is *0.I.2*. This rule has been built using the typical collocate of seize in its 0.I.2 sense, namely *initiative*. The translation associated to this sense number of *seize* in the dictionary is *prendre*, which

---

[4]Possibly translation, depending on the application

is the desired one in this context.

2.5.2 Rule competition

In some cases, many rules may apply to a given word in the same context, therefore we need a rule selection strategy.

Suppose one wants now to disambiguate the word *seize*, in the sentence:

The police seized a man employed by the Krugersdorp branch of the United Building Society on approximately 18 May 1985.

The dependencies extracted by the shallow parser which might lead to a disambiguation, i.e. which involve *seize*, are:

SUBJ(police,seize)
DOBJ(seize,man)
VMODOBJ(seize,about,1985)
VMODOBJ(seize,of,Society)
VMODOBJ(seize,by,branch)

In the case of our example, none of the rules of the database match directly the lexical context of the dependencies. Therefore, the system tries to match the semantic context of the dependency. To perform this task, the distance between the list of semantic classes of a potential rule (L1) and the list of semantic classes associated with the non-headword of the dependency (L2) is calculated:

$$d = \frac{(CARD(UNION(L1,L2)) - CARD(INTER(L1,L2)))}{CARD(UNION(L1,L2))}$$

To enable fast execution in terms of distance calculation, a transducer which associates a word with its WordNet top classes has been built and is loaded on the server. The distance calculated here ranges from 0 to 1, 0 meaning a full match of classes, 1 no match at all, the "best" rules being the ones with the smallest distance. In this particular example, the list of classes attached to *man* in WordNet is used to calculate the distance with the potential matching rules. Several rules now match the semantic context of the dependency DOBJ(seize,man).

After removing rules matching with a distance above some threshold, it appears that two potential matching rules still compete:

- one is built using the collocate *[prisoner]*: DOBJ(seize,prisoner) $\Rightarrow$ 0.I.3 <CO>;

at **class level** DOBJ(seize,18) $\Rightarrow$ 0.I.3 <CO>;

- the other is built using the example *to seize somebody around the waist*:
  DOBJ(seize,somebody) $\Rightarrow$ 0.I.1 <LE>;
  at **class level** DOBJ(seize,18) $\Rightarrow$ 0.I.1 <LE>;

Indeed, *prisoner* and *somebody* share the same semantic WordNet class (18, noun.animate) which is a member of the list of classes attached to *man* as well. The following preference strategy is applied[5]: first, prefer rules from collocate (<CO>), then from compounds examples (<LC>), then from structure examples (<LO>), then from phrasal verb pattern examples (<LV>), then from usage examples (<LU>), and then from general examples (<LE>). This strategy allows the selection of the rule to apply, here the one built with the collocate *[prisoner]*. The sense number attached by the system to *seize* is 0.I.3, the general meaning being *capture*, and the French translation *s'emparer de*.

In cases where two competing rules are exactly of the same type, the system chooses the first one (first sense appearing in the entry), relying on the fact that the OHFD was built using corpora: by default, semantic categories of the entries are ordered according to frequency in corpora.

2.5.3 Rule cooperation

The previous example showed how rules can compete between each other. But in some cases they can cooperate as well. Let's disambiguate *seize* in the following example sentence:

United States federal agents seized a surface-to-air rocket launcher, a rocket motor, range-finders and a variety of military manuals.

Since the sentence contains a coordinated direct object of *seize*, one gets the following dependencies from the shallow parser:

DOBJ(seize,launcher)
DOBJ(seize,motor)
DOBJ(seize,range-finder)
DOBJ(seize,manuals)

Many rules are matching at class level, with a given distance *d*, namely:

DOBJ(seize,4/6/11) $\Rightarrow$ 0.I.3 <CO>; d=0.75
DOBJ(seize,7/24/4/9/26/6/18/10) $\Rightarrow$ 0.I.3 <CO>; d=0.9
DOBJ(seize,8/6/14) $\Rightarrow$ 0.I.4 <CO> ; d=0.75
DOBJ(seize,21/7/15/9/6) $\Rightarrow$ 0.I.4 <CO> ; d=0.83

Two rules point out the sense number 0.I.3, the two others, the sense number 0.I.4. The strategy of rule selection takes this fact into account, giving more importance to sense numbers matching many times. As far as implementation is concerned, the distances associated with rules pointing on the same sense number are multiplied together. Since distances range from 0 to 1, multiplying them decreases the resulting value of the distance. Since the lowest one is chosen, the system put the emphasis on semantic redundancy. In the example, the distance finally associated with sense number 0.I.4 is 0.6625, which is smaller than the one associated with sense number 0.I.3 (0.675). The sense number selected by the system is therefore 0.I.4, the translation being *saisir*, which is the desired one. The same strategy is implemented for word level rules cooperation, in this case, rule weights are added.

2.6 Implementation

The different modules of the system presented here are implemented in C++ in the XeLDa client/server architecture:
- As already mentioned, the rule learner is a simple XeLDa client that performs rule extraction once ;
- The rule application program is implemented as a specific dictionary lookup service: when a word is semantically disambiguated with a rule, the application program reorders the dictionary entry according to the semantic category assigned to the word. The best matching part of the entry is then presented first. This application is built on top of Locolex (Bauer et al. 95), an intelligent dictionary lookup which achieves some word sense disambiguation using word context (part-of speech and multiword expressions (MWEs)[6] recognition). However, Locolex choices remain purely syntactic. Using the OHFD information about examples, collocates and subcategorization as well as semantic classes from an ontology, the system presented here goes further towards semantic disambiguation.

---

[5]At class level, idiomatic examples are not used, because the idiomatic expressions given in the dictionary are fully lexicalized

[6]Multiword expressions range from compounds (salle de bain) and fixed phrases (a priori) to idiomatic expressions (to sweep something under the rug).

3 Evaluation

We evaluated the system for English on the 34 words used in the SENSEVAL competition (Kilgarriff 98; Kilgarriff 99), as well as on the SENSEVAL corpus (HECTOR). This provided a test set of around 8500 sentences. The SENSEVAL words are all polysemous which means that the results given below reflect real polysemy.

We use the SENSEVAL test set for this in vitro evaluation in order to give us a mean of comparison, especially with the results obtained in this competition with GINGER II (Dini et al. 99). Still, it is important to keep in mind that this comparison is difficult since the dictionaries used are different. We used the OHFD bilingual dictionary while in SENSEVAL the Oxford monolingual dictionary from HECTOR was used.

The evaluation given below is performed if and only if the semantic disambiguator has found a matching rule, which means that the results focus only on our methodology: recall and precision would have been better if we had evaluated all outputs (even when the result is just the first meaning corresponding to the syntactic part of speech of the word in the sentence) because the OHFD gives by default the most frequent meaning of a word.

The results obtained with the system are given on the following table:

POS	Precision	Recall	Polysemy
N	83.7 %	27.4 %	5.4
A	81.3 %	55.8 %	5.7
V	75 %	37.6 %	6.2
Global	79.5 %	37.4 %	5.8

Numbers show that the recall is equivalent to the one we obtained with GINGER II (37.6 %) in SENSEVAL (this just means that dictionaries content is about the same) but precision is dramatically improved (46% for GINGER II for 79.5% with this system). Increase in precision is due to the fact that we used more fine-grained dictionary information. Moreover, the evaluation shows that the distribution of the precision results follows the preference strategy employed to select rules: collocate rules are more precise than examples rules, compounds or idiom rules are themselves more precise than usage examples, etc.

Another evaluation of smaller coverage has been performed on "all polysemous words" of about 400 sentences extracted from the *Times* newspaper, and shows similar results according to part of speech distribution.

POS	Precision	Recall	Polysemy
N	81 %	28.3 %	5.5
A	79 %	64 %	5.8
V	74 %	34.5 %	9.8
Global	78%	36.1 %	6.2

These results confirm that dictionary information is very reliable for semantic disambiguation tasks.

4 Conclusion and Future expectations

This paper describes a client/server implementation of a word sense disambiguator. The method uses a dictionary as a tagged corpus in order to extract a semantic disambiguation rule database. Since there is no need for a tagged training corpus, the method we describe, which performs "all words" semantic disambiguation, is unsupervised and avoids the data acquisition bottleneck observed in WSD. Rules are available to be used by a semantic application program which uses a specific linguistic strategy to select the best matching rule to apply: the rule selection is based on an SGML typed-based preference strategy and takes into account rules competition and rule cooperation.

Emphasis is put on the advantage of the client/server implementation in terms of robustness as well as on the good results provided by the strategy in terms of recall and precision. The client/server implementation provides robustness, modularity and fast execution.

The disambiguation strategy provides high precision results, because senses and examples have been defined by lexicographers and therefore provide a reliable linguistic source for constructing a database of semantic disambiguation rules. Recall results are good as well, meaning that the coverage of the dictionary is important.

These results could be improved by learning more disambiguation rules, for example using the correspondences between functional dependencies: when a dependency DOBJ(X,Y) is extracted, a rule for SUBJPASS(Y,X) can be built (and vice-versa). They could be improved as well by integrating more fine-grained semantic information for adverbs and adjective, WordNet being relatively poor for these parts of speech.

Since the architecture is modular, the system initially provided for English can be quickly adapted for any other language as soon as the required components are available. We already started to build a

semantic disambiguator for French, but we need to integrate a French semantic ontology into the system. At the moment, it is planned to extract such an ontology from the dictionary itself, using the semantic labels which are associated with semantic categories. The expectation is to obtain more consistency between semantic tags (dictionary) and semantic classes (ontology).

Because we used a bilingual dictionary we integrated the disambiguation module into a general system architecture dedicated to the comprehension of electronic texts written in a foreign language.

This technique coupled with other natural language processing techniques such as shallow parsing can also be used to extract general semantic networks from dictionaries or encyclopedia.

References

Elisabeth Aimelet. 1998. *XeLDa Dictionary Lookup Improvement* XRCE ATS XeLDa Technical Report.

S. Aït-Mokhtar, J-P. Chanod. 1997. Subject and Object Dependency Extraction Using Finite-State Transducers. In *Proceedings of the Workshop on automatic Information Extraction and the Building of Lexical Semantic Resources,, ACL*, p71-77, Madrid, Spain.

R. Akroyd. September 1992. Markup for the Oxford-Hachette French Dictionary, English to French. *Technical report* Oxford University Press.

D. Bauer, F. Segond, A. Zaenen. 1995. LOCOLEX: the translation rolls off your tongue. In *Proceedings of ACH-ALLC*, Santa-Barbara, USA.

L. Dini, V. Di Tomaso, F. Segond. 1998. Error Driven Word Sense Disambiguation In *Proceedings of COLING/ACL*, p320-324, Montreal, Canada.

L. Dini, V. Di Tomaso, F. Segond. 1999. GINGER II: an example-driven word sense disambiguator. In *Computer and the Humanities*, to appear.

C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*, MIT Press, Cambridge (MA).

W.A. Gale, K.W. Church, D. Yarowsky. 1992. Work on statistical methods for word sense disambiguation in *Probabilistic Approaches to Natural Language: Papers from the 1992 AAAI Fall Symposium*, p54-60, Cambridge, MA, October.

W.A. Gale, K.W. Church, D. Yarowsky. 1993. A method for dismbiguating word senses in a large corpus. in *Computer and the Humanities*, 26:415-439.

N. Ide. Véronis. 1990. Very large neural networks for word sense disambiguation. in *Proceedings of the 9th european conference on artificial intelligence, ECAI'90*, p. 366-368, Stockholm.

A. Kilgarriff. 1998. SENSEVAL: An Exercise in Evaluating Word Sense Disambiguation Programs. In *Proceeding of the First International Conference on Language Ressources and Evaluation*, Granada, Spain.

A. Kilgarriff. 1999. Gold standard datasets for evaluating word sense disambiguation programs. In *Computer and the Humanities*, to appear.

C. Leacock, G. Towell. 1993. Corpus-based statistical sense resolution. in *Proceedings of the ARPA Human Language technology workshop*, San Francisco, Morgan Kaufman.

H.T. Ng, H.B. Lee. 1996. Integrating Multiple Knowledge Sources to Disambiguate Word Sense: an Examplar-based Approach. In *Proceedings of the ACL*, p.40-47.

Oxford-Hachette. 1994. *The Oxford Hachette French Dictionary*. Edited by M.-H. Corréard and V. Grundy, Oxford University Press-Hachette.

P. Resnik and D. Yarowsky. 1997. A perspective on word sense disambiguation methods and their evaluation. In *Proceedings of ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?*, Washington D.C., USA.

Claude Roux. 1998. *XeLDa Shallow Parser* XRCE ATS XeLDa Technical Report.

J. Véronis, N. Ide. 1998. Introduction to the Special Issue on Word Sense Disambiguation: The State of the Art. In *Computational Linguistics* 24/1.

J. Véronis, N. Ide. 1990. Word sense disambiguation with very large neural networks extracted from very large corpora In *Proceedings of the 13th international conference on computational linguistics, COLING'90*, volume 2, p.389-394, Helsinki, Finland.

D. Yarowsky. 1995. Unsupervised word sense disambiguation method rivalizing supervised methods. In *Proceedings of the ACL*, p189-196.

Y. Wilks, M. Stevenson. 1998. Word Sense Disambiguation using Optimised Combinations of Knowledge Sources. In *Proceedings of COLING/ACL*, Montreal, Canada.