

## A NEW ALGORITHM FOR WORD SENSE DISAMBIGUATION

DOINA TĂȚAR AND GABRIELA ȘERBAN

**ABSTRACT.** The task of disambiguation is to determine which of the senses of an ambiguous word is invoked in a particular use of the word [3]. Starting from the algorithm of Yarowsky [5, 4] and the Naive Bayes Classifier (NBC) algorithm, in this paper we propose an original algorithm which combines their elements. This algorithm preserve the advantage of principles of Yarowsky (*one sense per discourse and one sense per collocation*) with the known high performance of the NBC algorithm. Moreover, an agent is constructed accomplishing this algorithm.

### 1. INTRODUCTION

The word sense disambiguation (WSD) is probably one of the most important open problem and it has now already a long "history" in computational linguistics [2, 1]. WSD problem has direct applications in some fields of text understanding as *information retrieval, text summarization, machine translation*.

The problem that arises in word sense disambiguation (WSD) in natural language is that many words (called polysemic), have several meanings or senses. These senses depend on the context they occur in. The task of disambiguation is to determine which of the senses of an ambiguous word is invoked in a particular use of the word [3]. Whenever a system's actions depend on the meaning of the text being processed, WSD is necessary.

The algorithms used in WSD are classified considering whether they involve supervised or unsupervised learning. Unsupervised learning can be viewed as clustering task while supervised learning is usually seen as a classification task. Dictionary based disambiguation, which we will present in the following section can be considered as intermediary between supervised and unsupervised disambiguation [3, 6].

---

2000 *Mathematics Subject Classification.* 68T50, 68Q32.

1998 *CR Categories and Descriptors.* 1.2.7[**Computing Methodologies**]: Artificial Intelligence – *Natural Learning Processing*; 6.3[**Mathematicsof Computing**]: Statistical Computing – .

## 2. DICTIONARY BASED DISAMBIGUATION

If we have no information about the senses of a target word  $w$  we can follow disambiguation methods that rely on the definitions in dictionaries.

Notational conventions used in the following are:

- $w$ — the word to be disambigued ( *target word*);
- $s_1, \dots, s_K$ — possible senses for  $w$ ;
- $c_1, \dots, c_I$ — contexts of  $w$  in corpus;
- $v_1, \dots, v_J$ — words used as contextual features for disambiguation of  $w$ .

Regarding of  $v_1, \dots, v_J$  there are two possibilities: they are collocates or co-occurrences with  $w$ . In the first case the contextual features occur in a fixed position near  $w$ , in a *window* of fixed length, centered on  $w$ . In the second case the contextual features occur together with  $w$ , in arbitrarily positions. We will consider the first sense of contextual features.

In [5] (1995), Yarowsky observed that there are constraints between different occurrences of contextual features that can be used for disambiguation. Two such constraints are:

- *One sense per discourse*: the sense of a target word is highly consistent within a given discourse (document);
- *One sense per collocation*: the contextual features (nearby words) provide strong clues to the sense of a target word.

For example, regarding the first constraint for the word *plant*, if his sense is in a first occurrence "living being", then later occurrences are likely to refer to "living beings" too. As the second constraint for *plant*, if the word *animal* occurs together with *plant*, this word is likely to be a clue word for the "living beings" sense.

The algorithm proposed by Yarowsky combines both constraints. It iterates building two sets ,  $F_k$  and  $E_k$  for each sense  $s_k$ :  $F_k$  contains characteristic collocations,  $E_k$  is the set of contexts of the target word  $w$  which are assigned to the sense  $s_k$ . The algorithm is as bellow [3]. Let us remark that a multi-set is denoted by  $\{\{\dots\}\}$ :

*Initialization*

**for all sense  $s_k$  of  $w$  do**

$F_k = \{\text{the set of collocations in definition from}$   
*dictionary of  $s_k$  sense of  $w$*

**for all  $s_k$  of  $w$  do**

$E_k = \Phi$

*One sense per collocation*

**While** at least one  $E_k$  changed in the last iteration **do**  
     **for** all sense  $s_k$  of  $w$  **do**  
          $E_k = \{\{c_i \mid c_i \cap F_k \neq \Phi\}\}$   
     **for** all sense  $s_k$  of  $w$  **do**  
          $F_k = \{f_m \mid \forall n \neq k, \frac{P(s_k|f_m)}{P(s_n|f_m)} > \alpha \text{ (usually } \alpha = 1)\}$

*One sense per discourse*  
**for** all document  $d_m$  (context or set of contexts) **do**  
     *determine the majority sense  $s_k$  of  $w$  in  $d_m$*   
     *assign all occurrences of  $w$  in  $d_m$  to  $s_k$*

### 3. SUPERVISED DISAMBIGUATION BY NAIVE BAYES CLASSIFIER ALGORITHM

In supervised disambiguation a tagged corpus or a semantic annotated corpus is available. Such annotated corpus is used in on-line product Senseval. The task in this case is to build a classifier which correct classifies a new context based on the contextual features occurring in this context. The classifier does no feature selection, but it combines the participation of all contextual features.

What a Naive Bayes Classifier realizes is the calculus of the sense  $s'$  which for the target word  $w$  and a given context  $c$  satisfies the relation:

$$(1) \quad s' = \underset{s_k}{\operatorname{argmax}} P(s_k \mid c) = \underset{s_k}{\operatorname{argmax}} \frac{P(c \mid s_k)}{P(c)} P(s_k) \\ = \underset{s_k}{\operatorname{argmax}} P(c \mid s_k) P(s_k).$$

The same value for  $s'$  is obtained if we consider the logarithm of expression:

$$(2) \quad s' = \underset{s_k}{\operatorname{argmax}} (\log P(c \mid s_k) + \log P(s_k))$$

The Naive Bayes assumption is that the contextual features are all conditional independent:

$$(3) \quad P(c \mid s_k) = P(\{v_j \mid v_j \in c\} \mid s_k) = \prod_{v_j \in c} P(v_j \mid s_k).$$

Here  $v_j$  represents any word in the context  $c$ .

This assumption has two consequences:

- the structure and order of words in context is ignored;
- the presence of one word in the context does not depend on the presence of another.

This is clearly not true, but there is a large number of cases in which the algorithm works well.

As regarding the probabilities  $P(v_j | s_k)$  and  $P(s_k)$ , these are calculated from the labeled (annotated) corpus:

$$(4) \quad P(v_j | s_k) = \frac{C(v_j, s_k)}{C(s_k)} \quad P(s_k) = \frac{C(s_k)}{C(w)}$$

where  $C(v_j, s_k)$  is the number of occurrences of  $v_j$  in the contexts annotated with the sense  $s_k$ ,  $C(s_k)$  is the number of contexts with the sense  $s_k$  and  $C(w)$  is the total number of occurrences of the word  $w$ .

The NBC algorithm is:

*Training:*

```

for all senses  $s_k$  of  $w$  do
    for all words  $v_j$  in corpus) do
         $P(v_j | s_k) = \frac{C(v_j, s_k)}{C(s_k)}$ 
for all senses  $s_k$  of  $w$  do
     $P(s_k) = \frac{C(s_k)}{C(w)}$ 

```

*Disambiguation:*

```

for all senses  $s_k$  of  $w$  do
     $score(s_k) = \log P(s_k) + \sum_{v_j \in c} \log P(v_j | s_k)$ 
Calculate  $s' = \operatorname{argmax}_{s_k} score(s_k)$ 

```

In [3] is reported that a disambiguation system based on this algorithm is correct for about 90 percents of cases.

#### 4. A BOOTSTRAPPING ALGORITHM ON THE BASE OF THE PRINCIPLES: ONE SENSE PER DISCOURSE AND ONE SENSE PER COLLOCATION

The algorithm begins by identifying a small number of training contexts. This could be accomplished by hand tagging with senses the contexts of  $w$  for which the sense of  $w$  is clear because some *seed collocations* [5] occur in these contexts.

This tagging is made on the base of dictionaries or by using the known on-line dictionary of senses WordNet . This initial set of annotated contexts is used for learning a *naive bayesian classifier*. This NBC will help in annotating new contexts. By repeating the process, the annotated part of corpus grows. We will

stop when the remaining unannotated corpus is empty or any new context can't be annotated.

The notational conventions are as above:

- $w$  is the polysemic word
- $S = \{s_1, s_2, \dots, s_K\}$  are possible senses for  $w$ , as in a dictionary, or as obtained with WordNet.
- $C = \{c_1, c_2, \dots, c_I\}$  are contexts (windows) for  $w$ , as obtained for  $w$  with an on-line corpus tool ( for example Cobuild). each  $c_i$  is of the form:

$$c_i = w_1, w_2, \dots, w_t, w, w_{t+1}, \dots, w_z$$

where  $w_1, w_2, \dots, w_t, w_{t+1}, \dots, w_z$  are words from the set  $v_1, \dots, v_J$  and  $t$  and  $z$  ( usually  $z = 2t$ ) are selected by user.

Let us consider that the words  $V = \{v^1, \dots, v^l\} \subset \{v_1, \dots, v_J\}$ , where  $l$  is small ( for example 2 ) are *surely* associated with the senses for  $w$ , such that the occurrence of  $v^i$  in the context of  $w$  determines the choice for  $w$  of a sense (one sense per collocation).

For example, for the word *plant*, the occurrence in the same context of the word *life* means a sense ( let say A ) , while the occurrence in the same context of the word *manufacturing* means another sense ( let say B). These rules can be done as a decision list:

$$\text{if } v^i \text{ occurs in a context of } w \text{ (of } z \text{ words)} \Rightarrow s^i, i = 1, \dots, l$$

So, some contexts can be solved from the set of contexts obtained as query results with Cobuild. Namely, we marked these contexts with A or B:

- (A)industrial equipment and engineering plant.[p] The company insures
- (A)hard currency. And so we've found a plant, and I have some seeds here from
- (B)the planning and construction of the plant at Rabta near Tripoli and were
- (A)A. japonica Japanese aucuba. A male plant, bearing panicles of purple-
- (A)and experience of any individual plant in my garden alone is hardly
- (A)aspect, features and animal and plant life."[p] [p] These were never
- (A)in flower and it Is worth having a plant or two in the flower border or in
- (B)all the allegations. It says the plant produces merely pharmaceuticals.
- (A)or example, the issue of the role of plant respiration in

- a hydrological
- (A)he USA announced in 1989 that 680 US plant species will  
be extinct in the wild
- (B)d be looking at 75 to 100 jobs and a plant that would  
produce probably
- (B)the Sellafield nuclear reprocessing plant. These are ‘cost  
plus" contracts
- (B)[h] [p] SCIENTISTS have engineered a plant which could  
grow its own plastic

### Algorithm

We start by defining a relation  $\delta : WX C$ , where  $W$  is the set of words and  $C$  is the set of contexts (set of array of words). If  $w \in W$  is a word and  $c \in C$  is a context, we say that  $(w, c) \in \delta$  if exist a word  $w_1 \in c$  so that the words  $w$  and  $w_1$  have the same root.

$C_{res} = \Phi$ , determine the set  $V = \{v^1, \dots, v^l\}$

**For** each context  $c$  in  $C$  apply the rules:

**if**  $(v^i, c) \in \delta, \Rightarrow$  sense  $s^i, i = 1, \dots, l, C_{res} = C_{res} \cup \{c\}$

$C_{rest} = C \setminus C_{res}$

**While**  $C_{rest} \neq \Phi$  **do** :

*Determine a set  $V^*$  of words with maxim frequency in  $C_{res}$*

*Define  $V = V \cup V^* = \bigcup_{j=1}^l V_{s_j}$ ,*

*where  $V_{s_j}$  is the set of words associate with the sense  $s_j$*

*(If  $v \in V^*$ , the context  $c$  solved with the sense  $s_j$ , and  $(v, c) \in \delta$ ,*

*then  $v \in V_{s_j}$ , according with the principle ‘one sense per discourse’)*

***For** each  $c_i \in C_{rest}$  apply the BNC algorithm :*

$$(5) \quad s_i^* = \operatorname{argmax}_s P(s | c_i) = \operatorname{argmax}_s \frac{P(c_i | s) \times P(s)}{P(c_i)}$$

$$= \operatorname{argmax}_s P(c_i | s) \times P(s)$$

where  $P(c_i | s) = P(w_1 | s) \cdots P(w_t | s) P(w_{t+1} | s) \cdots P(w_z | s)$

$$\text{and } P(w_i | s_j) = \begin{cases} 1 & \text{if } w_i \in V_{s_j} \\ \frac{\text{nr.occ. } w_i}{\text{nr. total of words}} & \text{else} \end{cases}$$

$$C_{res}^* = \{c_i | P(s_i^* | c_i) > N, N \text{ fixed}\}$$

$$C_{res} = C_{res}^* \cup C_{res}$$

$$C_{rest} = C_{rest} \setminus C_{res}$$

## 5. THE APPLICATION

**5.1. The Agent for words’ disambiguation. General presentation.** The application is written in Visual C++ 6.0 (Figure 1) and implements the behavior

of an Intelligent Agent, whose purpose is to find the correct sense for a given word (the target word) in some given contexts (the word's disambiguation), using the algorithm described in the previous section.

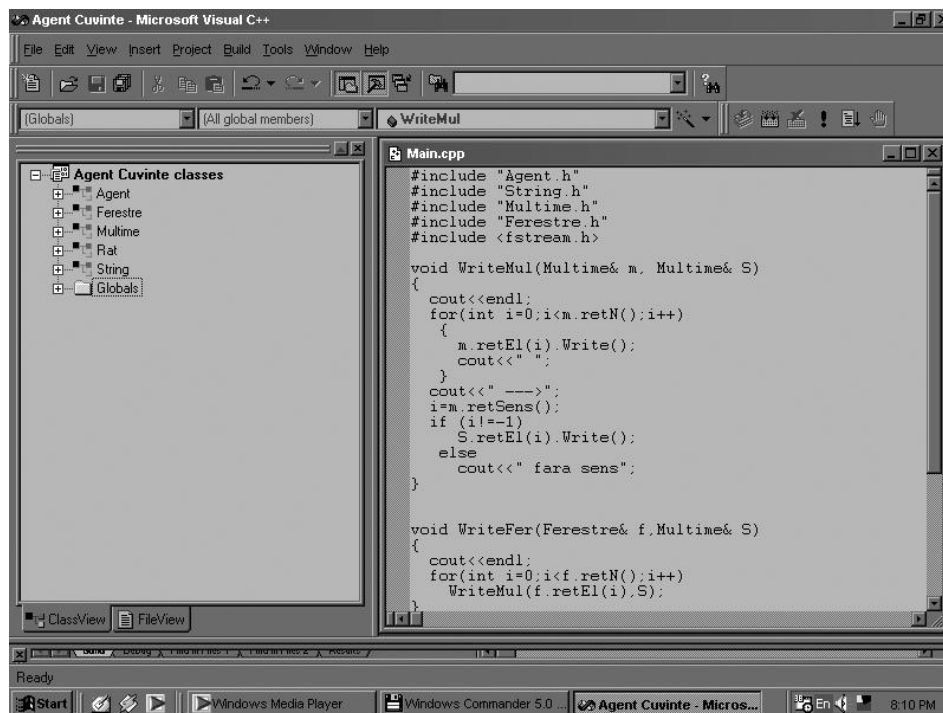


FIGURE 1. The Agent

The environment of this agent consists in some information which the agent reads from an input text file "in.txt":

- the target word( $w$ );
- the possible senses for  $w$ ;
- the contexts for  $w$ ;
- the words used as contextual attributes for  $w$ 's disambiguation.

On the basis of his environment, using the algorithm described in the previous section, the agent learns to find the correct sense of the target word in the given contexts.

5.2. **The Agent's design.** The basis classes used for implementing the agent's behavior are the following:

- **String:** defines the type *String* (array of characters), having methods for:
  - adding a char in a *String*;
  - accessing the length and the characters of a *String*;
  - displaying, comparing, concatenating *Strings*.
- **Set:** defines the type *Array* of strings (corresponding to a context which contains the target word  $w$ ), associated with a sense of  $w$ . The main methods of this class are for:
  - adding a *String* in an *Array*;
  - accessing the number of elements and the strings of an *Array*;
  - testing the membership of a string in the *Array*;
  - setting the corresponding sense for  $w$ ;
  - finding the reunion of two *Arrays*.
- **Contexts:** defines the type *Set* of arrays of strings (array of contexts), representing the contexts for which we want to associate a sense corresponding to  $w$ . The main methods of this class are for:
  - adding an element in the *Set*;
  - accessing the number of elements and the elements of a *Set*;
  - testing the membership of an array in the *Set*;
  - finding the difference of two *Sets*.
- **Agent:** the main class of the application, which implements the agent behavior and the learning algorithm (Figure 2).  
The private member data of this class are:
  - **Q:** the target word;
  - **S:** the set of senses for the target word;
  - **v:** the set of words used as contextual attributes for  $Q$ 's disambiguation;
  - **C:** the contexts for the target word.

The public methods of the agent are the followings:

- **readEnvironment:** reads the information about the environment from an input stream ;
- **disambiguation:** the main (learning)algorithm of the agent used to find the correct senses of the target word in the environment's contexts;
- **retQ:** returns the target word ( $Q$ );
- **retS:** returns the set of senses of the target word ( $S$ );
- **retV:** returns the member data  $v$ ;
- **retC:** returns the contexts for the target word ( $C$ );

Besides the public methods, the agent has some private methods used in the method **disambiguation**.

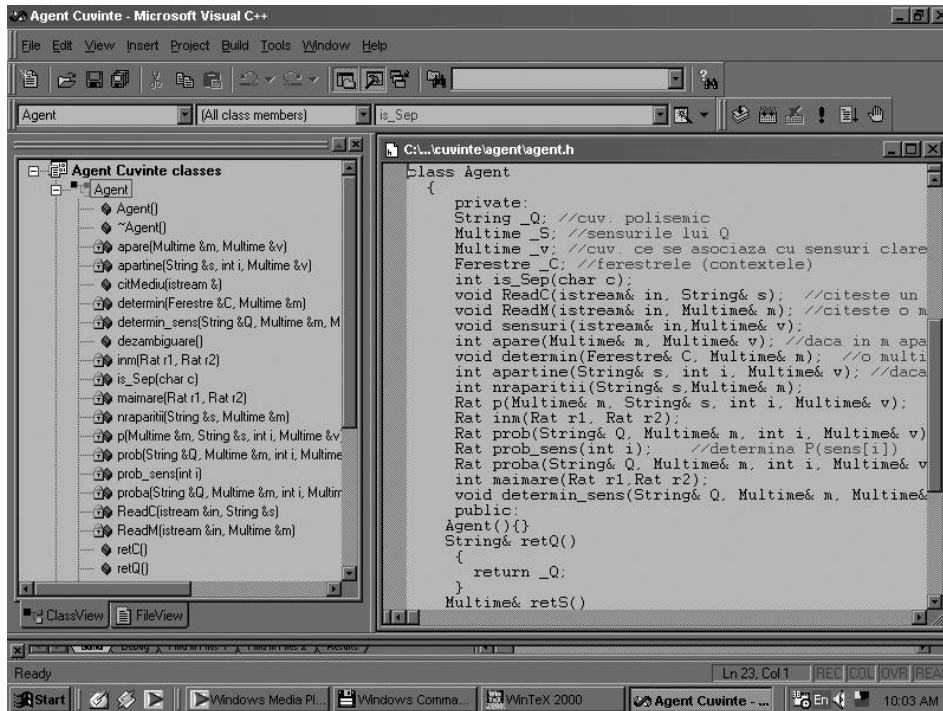


FIGURE 2. The main class of the Agent

5.3. **Experiment.** Using the application we accomplish the training of the agent in the following environment (given in the text file "in.txt"):

---

<b>poarta</b>	the target word
<b>usa imbraca</b>	the senses of the target word
<b>clanta 1 se 2</b>	the words used as contextual attributes for $Q$ 's disambiguation and the indexes of the corresponding sense of the target word
<b>poarta se deschide cu clanta</b>	the contexts of the target word
<b>se poarta rosu</b>	
<b>unde este poarta</b>	
<b>se stie ca cine poarta rosu este optimist</b>	
<b>daca poarta e inchisa nu intram</b>	

---

After the agent reads the information from the environment, he applies the disambiguation algorithm for the given contexts. The result is shown below (each context is followed by the sense for the target word - found by the agent after the disambiguation).

---

*poarta se deschide cu clanta — usa*  
*se poarta rosu — imbraca*  
*unde este poarta — usa*  
*se stie ca cine poarta rosu este optimist — imbraca*  
*daca poarta e inchisa nu intram — usa*

---

We notice that if the Agent starts with a substantial initial knowledge (number of senses of the target word, set of words used as contextual attributes for the disambiguation) and if the environment consists in a big number of contexts, the the disambiguation (learning) algorithm works very well.

#### REFERENCES

- [1] J. Allen : " Natural language understanding", Benjamin/Cummings Publ. , 2nd ed., 1995.
- [2] D. Jurafsky, J. Martin: " Speech and language processing", Prentice Hall, 2000.
- [3] C. Manning, H. Schutze: " Foundation of statistical natural language processing", MIT, 1999.
- [4] P. Resnik, D. Yarowsky : " Distinguishing Systems and Distinguishing sense: new evaluation methods fot WSD ", Natural Language Engineering, 1 , nr 1, 1998.
- [5] D. Yarowsky: " Hierarchical Decision Lists for WSD", Kluwer Acadmic Publishers, 1999.
- [6] F. Sebastiani: " A tutorial on Automated Text Categorization", pp 1-25, ESLLI 2001.
- [7] D. Tatar: " Inteligenta artificiala: demonstrare automata de teoreme, prelucrarea limbajului natural", Editura Microinformatica, 2001.

UNIVERSITY " BABEȘ-BOLYAI", CLUJ-NAPOCA, ROMANIA  
*E-mail address: dtatar@cs.ubbcluj.ro, gabis@cs.ubbcluj.ro*