

Word Sense Disambiguation (WSD)

Based on

“Foundations of Statistical NLP” by C. Manning & H. Schütze, ch. 7
MIT Press, 2002

WSD Examples

They have the **right** to bear arms. (*drept*)

The sign on the **right** was bent. (*direcție*)

The **plant** is producing far too little to sustain its operation for more than a year. (*fabrică*)

An overabundance of oxygen was produced by the **plant** in the third week of the study. (*plantă*)

The **tank** has a top speed of 70 miles an hour, which it can sustain for 3 hours. (*tanc petrolier*)

We cannot fill more gasoline in the tank. (*rezervor de mașină*)

The **tank** is full of soldiers. (*tanc de luptă*)

The **tank** is full of nitrogen. (*recipient*)

Plan

1. Supervised WSD

1.1 A Naive Bayes Learning algorithm for WSD

1.2 An Information Theoretic algorithm for WSD

2. Unsupervised WSD clustering

2.1 WS clustering: the EM algorithm

2.2 Constraint-based WSD:

“One sense per discourse, one sense per collocation”: Yarowsky’s algorithm

2.3 Resource-based WSD

2.3.1 Dictionary-based WSD: Lesk’s algorithm

2.3.2 Thesaurus-based WSD:

Walker’s algorithm

Yarowsky’s algorithm

1.1 Supervised WSD through Naive Bayesian Classification

$$\begin{aligned}
 s' &= \operatorname{argmax}_{s_k} P(s_k | c) = \operatorname{argmax}_{s_k} \frac{P(c|s_k)}{P(c)} P(s_k) = \\
 &\operatorname{argmax}_{s_k} P(c | s_k) P(s_k) = \operatorname{argmax}_{s_k} [\mathbf{log} P(c | s_k) + \mathbf{log} P(s_k)] = \\
 &\operatorname{argmax}_{s_k} [\mathbf{log} P(s_k) + \sum_{w_j \text{ in } c} \mathbf{log} P(w_j | s_k)]
 \end{aligned}$$

where we used the naive Bayes assumption:

$$P(c | s_k) = P(\{w_j | w_j \text{ in } c\} | s_k) = \prod_{w_j \text{ in } c} P(w_j | s_k)$$

The Maximum Likelihood estimation:

$$P(w_j | s_k) = \frac{C(w_j, s_k)}{C(s_k)} \quad \text{and} \quad P(s_k) = \frac{C(w, s_k)}{C(w)} \quad \text{where:}$$

$C(w_j, s_k)$ = number of occurrences of word w_j with the sense s_k

$C(w, s_k)$ = number of occurrences of word w with the sense s_k

$C(w)$ = number of occurrences of the ambiguous word w

all counted in the training corpus.

A Naive Bayes Algorithm for WSD

```
1  comment: training
2  for all senses  $s_k$  of  $w$  do
3    for all words  $w_j$  in the vocabulary do
4      
$$P(w_j|s_k) = \frac{C(w_j, s_k)}{C(w_j)}$$

5    end
6  end
7  for all senses  $s_k$  of  $w$  do
8    
$$P(s_k) = \frac{C(w, s_k)}{C(w)}$$

9  end
10 comment: Disambiguation
11 for all senses  $s_k$  of  $w$  do
12   score( $s_k$ ) = log( $P(s_k)$ )
13   for all words  $w_j$  in the context window  $c$  do
14     score( $s_k$ ) = score( $s_k$ ) + log( $P(w_j|s_k)$ )
15   end
16 end
17 choose  $s' = \operatorname{argmax}_{s_k} \operatorname{score}(s_k)$ 
```

1.2 An Information Theoretic approach to WSD

Remark:

The Naive Bayes classifier attempts to use information from words in the context window to help in the disambiguation decision.

It does this at the cost of a somewhat unrealistic independence assumption.

The IT (“Flip-Flop”) algorithm that follows does the opposite:

It tries to find a single contextual feature that reliably indicates which sense of the ambiguous word is being used.

Empirical result: The Flip-Flop algorithm improved by 20% the accuracy of a Machine Translation system.

Example

Highly informative indicators for 3 ambiguous French words

Ambiguous word	Indicator	Examples:
prendre	object	value → sense <i>mesure</i> → <i>to take</i> <i>décision</i> → <i>to make</i>
vuloir	tense	present → <i>to want</i> conditional → <i>to like</i>
cent	word to the left	<i>per</i> → % number → <i>c.</i> [money]

Notations

t_1, \dots, t_m — translations of the ambiguous word
example: *prendre* → *take, make, rise, speak*

x_1, \dots, x_n — possible values of the indicator
example: *mesure, décision, example, note, parole*

Mutual Information:

$$I(X; Y) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$$

Note: The Flip-Flop algorithm only disambiguates between 2 senses. For the more general case see [Brown et al., 1991a].

Brown et al.'s WSD (“Flip-Flop”) algorithm:

Finding indicators for disambiguation

```
1 find random partition  $P = \{P_1, P_2\}$  of  $t_1, \dots, t_m$ 
2 while (improving  $I(P; Q)$ ) do
3   find partition  $Q = \{Q_1, Q_2\}$  of  $x_1, \dots, x_n$ 
4     that maximizes  $I(P; Q)$ 
5   find partition  $\{P_1, P_2\}$  of  $t_1, \dots, t_m$ 
6     that maximizes  $I(P; Q)$ 
7 end
```

“Flip-Flop” algorithm

Note: Using the splitting theorem, [Breiman et al., 1984], it can be shown that the Fkflip-Flop algorithm monotonically increases $I(P; Q)$.

Stopping criterion:

$I(P; Q)$ does not increase anymore (significantly).

“Flip-Flop” algorithm:
Disambiguation

For the occurrence of the ambiguous word,
determine the value x_i of the indicator;

if $x_i \in Q_1$ then assign the occurrence the sense 1;

if $x_i \in Q_2$ then assign the occurrence the sense 2.

A running example

1. a randomly taken partition $P = \{P_1, P_2\}$:

$$P_1 = \{take, rise\}, P_2 = \{make, speak\}$$

2. maximizing $I(P; Q)$ on Q , using the (presumable) data

take a measure
 notes
 an example
 make a decision
 a speech
 rise to speak

$$Q_1 = \{measure, note, example\}, Q_2 = \{d\u00e9cision, parole\}$$

3. maximizing $I(P; Q)$ on P :

$$P_1 = \{take\}, P_2 = \{make, rise, speak\}$$

Note: Consider more than 2 ‘senses’ to distinguish between $\{make, rise, speak\}$.

2. Unsupervised word sense clustering

2.1 The EM algorithm

Notation:

K is the number of desired senses;

c_1, c_2, \dots, c_I are the contexts of the ambiguous word in the corpus;

w_1, w_2, \dots, w_J are the words being used as disambiguating features.

Parameters of the model (μ):

$$P(w_j | s_k), 1 \leq j \leq J, 1 \leq k \leq K$$

$$P(s_k), 1 \leq k \leq K.$$

Given μ , the **log-likelihood of the corpus C** is computed as:

$$l(C | \mu) = \log \prod_{i=1}^I P(c_i) = \log \prod_{i=1}^I \sum_{k=1}^K P(c_i | s_k) P(s_k) = \\ \sum_{i=1}^I \log \sum_{k=1}^K P(c_i | s_k) P(s_k)$$

Note: To compute $P(c_i | s_k)$, use the Naive Bayes assumption:

$$P(c_i | s_k) = \prod_{w_j \text{ in } c_i} P(w_j | s_k).$$

Procedure:

1. Initialize the parameters of the model μ randomly.
2. While $l(C | \mu)$ is improving, repeat
 - a. **E-step:** estimate the (posterior) probability that the sense s_k generated the context c_i :

$$h_{ik} = \frac{P(c_i | s_k)}{\sum_{k=1}^K P(c_i | s_k)}$$

- b. **M-step:** re-estimate the parameters $P(w_j | s_k)$, $P(s_k)$, by way of MLE:

$$P(w_j | s_k) = \frac{\sum_{\{c_i | w_j \text{ in } c_i\}} h_{ik}}{Z_k} \quad \text{and} \quad P(s_k) = \frac{\sum_{i=1}^I h_{ik}}{\sum_{k=1}^K \sum_{i=1}^I h_{ik}} = \frac{\sum_{i=1}^I h_{ik}}{I}$$

where Z_k is a normalizing constant:

$$Z_k = \sum_{k=1}^K \sum_{\{c_i | w_j \text{ in } c_i\}} h_{ik}.$$

2.2 Constraint-based WSD:

“One sense per discourse, one sense per collocation”

[Yarowsky, 1995]

One sense per discourse: The sense of a target word is highly consistent within any given document.

One sense per collocation: Nearby words provide strong and consistent clues to the sense of a target word, conditional on relative distance, order, and syntactic relationship.

Yarowsky Algorithm: WSD by constraint propagation

```

1  comment: Initialization
2  for all senses  $s_k$  of  $w$  do
3       $F_k =$  the set of features (words) in  $s_k$  dictionary definition
4       $E_k = \emptyset$ 
5  end

6  comment: one sense per discourse
7  while (at least one  $E_k$  changed in the last iteration) do
8      for all senses  $s_k$  of  $w$  do
9          comment: identify the contexts  $c_i$  bearing the sense  $s_k$ 
10          $E_k = \{c_i \mid \exists f_m \in F_k : f_m \in c_i\}$ 
11     end
12     for all senses  $s_k$  of  $w$  do
13         comment: retain the features  $f_m$  which best express sense  $s_k$ 
14          $F_k = \{f_m \mid \forall n \neq k \frac{P(s_k|f_m)}{P(s_n|f_m)} > \alpha\}$  where  $P(s_i|f_m) = \frac{C(f_m, s_i)}{\sum_j C(f_m, s_j)}$ 
15     end
16 end

```

Yarowsky Algorithm (Cont'd)

17 comment: one sense per collocation

18 determine the majority sense s_k of w in the document d :

19 $s_k = \operatorname{argmax}_{s_i} P(s_i)$ where $P(s_i) = \frac{\sum_{m \in F_i} C(f_m, s_i)}{\sum_j \sum_{m \in F_i} C(f_m, s_j)}$

20 assign all occurrences of w in the document d the sense s_k

21 end

2.3 Resource-based WSD

2.3.1 Dictionary-based WSD

Example

Two senses of *ash*:

sense	definition
s_1 : tree	D_1 : a tree of the olive family
s_2 : burned stuff	D_2 : the solid residue left when combustible material is burned

Disambiguation of *ash* using Lesk's algorithm (see next slide):

scores	context
s_1 s_2	
0 1	This cigar burns slowly and creates a stiff <i>ash</i> .
1 0	The <i>ash</i> is one of the last trees to come into leaf.

Dictionary-based WSD: Lesk's algorithm

```
1  comment: training
2  for all senses  $s_k$  of  $w$  do
3     $\text{score}(s_k) = \text{overlap}( D_k, \cup_{v_j \text{ in } c} E_{v_j} )$ 
4  end
5  comment: Disambiguation
6  choose  $s' = \text{argmax}_{s_k} \text{score}(s_k)$ 
```

where:

D_k is the set of words occurring in the dictionary definition of the sense s_k for w ;

E_{v_j} is the set of words occurring in the dictionary definition of v_j (the union of all its sense definitions)

2.3.2 Thesaurus-based WSD

Example

Word	Sense	Thesaurus category (Roget)
<i>bass</i>	musical senses	MUSIC
	fish	ANIMAL, INSECT
<i>star</i>	space object	UNIVERSE
	celebrity	ENTERTAINER
	star shaped object	INSIGNIA
<i>interest</i>	curiosity	REASONING
	advantage	INJUSTICE
	financial	DEBT
	share	PROPERTY

Thesaurus-based WSD: Walker's algorithm

```

1  comment: Given: context  $c$ 
2  for all senses  $s_k$  of  $w$  do
3     $\text{score}(s_k) = \sum_{w_j \text{ in } c} \delta(t(s_k), w_j)$ 
1  comment: score = #words compatible with the category of  $s_k$ 
5  end
6  comment: Disambiguation
7  choose  $s' = \text{argmax}_{s_k} \text{score}(s_k)$ 

```

where:

$t(s_k)$ is the thesaurus category of the sense s_k ;

$\delta(t(s_k), w_j) = 1$ iff $t(s_k)$ is one of the thesaurus categories for w_j and 0 otherwise.

Thesaurus-based WSD: Yarowsky's algorithm

```

1  comment: categorize words/topics on contexts  $c$ 
2  for all words  $w_j$  in the vocabulary do
3     $C_j = \{c \mid w_j \text{ in } c\}$ 
4  end
5  for all topics  $t_l$  do
6     $T_l = \{c \mid t_l \in t(c)\}$ 
7  end
8  for all words  $w_j$  and all topics  $t_l$  do
9     $P(w_j \mid t_l) = |C_j \cap T_l| / \sum_j |C_j \cap T_l|$ 
10 end
11 for all topics  $t_l$  do
12    $P(t_l) = (\sum_j |C_j \cap T_l|) / (\sum_l \sum_j |C_j \cap T_l|)$ 
13 end
14 comment: Disambiguation
15 for all senses  $s_k$  of  $w$  occurring in  $c$  do
16    $\text{score}(s_k) = \log P(t(s_k)) + \sum_{w_j \text{ in } c} \log P(w_j \mid t(s_k))$ 
17 choose  $s' = \operatorname{argmax}_{s_k} \text{score}(s_k)$ 

```

Addenda: The t -Test for Comparing the Performance of two Systems

1. Compute the mean μ_i and the variance s_i^2 , when dividing the data into n parts.
2. compute the t -value $t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{s^2}{n}}}$, with $s^2 = \frac{s_1^2 + s_2^2}{n-1}$.
3. Find C , the confidence level corresponding to the above-computed t -value in the table, considering $2(n-1)$ degrees of freedom.

See

[Diettrich, 1998] for a systematic discussion,
[Mooney, 1996], a case study for Word Sense Disambiguation.