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. (*direţie*)

The plant is producing far too little to sustain its operation for more than a year. (*fabrică*)
An overboundance 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 maşină*)
The tank is full of soldiers. (*tank 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

$$s' = argmax_{s_k} P(s_k \mid c) = argmax_{s_k} \frac{P(c|s_k)}{P(c)} P(s_k) =$$
$$argmax_{s_k} P(c \mid s_k) P(s_k) = argmax_{s_k} [\boldsymbol{log}\, P(c \mid s_k) + \boldsymbol{log}\, P(s_k)] =$$
$$argmax_{s_k} [\boldsymbol{log}\, P(s_k) + \Sigma_{w_j \text{ in } c}\ \boldsymbol{log}\, P(w_j \mid s_k)]$$

where we used the naive Bayes assumption:

$$P(c \mid s_k) = P(\{w_j \mid w_j \text{ in } c\} \mid s_k) = \Pi_{w_j \text{ in } c}\ P(w_j \mid s_k)$$

The Maximum Likelihood estimation:

$P(w_j \mid s_k) = \frac{C(w_j, s_k)}{C(s_k)}$ **and** $P(s_k) = \frac{C(w, s_k)}{C(w)}$ **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

comment: training
for all senses s_k of w do
  for all words w_j in the vocabulary do
    $P(w_j|s_k) = \frac{C(w_j,s_k)}{C(w_j)}$
  end
end
for all senses s_k of w do
  $P(s_k) = \frac{C(w,s_k)}{C(w)}$
end
comment: Disambiguation
for all senses s_k of w do
  score$(s_k) = \log(P(s_k))$
  for all words w_j in the context window c do
    score$(s_k) =$ score$(s_k) + \log(P(w_j|s_k))$
  end
end
choose $s' = \text{argmax}_{s_k}$ 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: value → sense
prendre	**object**	***mesure* → *to take*** ***décision* → *to make***
vuloir	**tense**	**present → *to want*** **conditional → *to like***
cent	**word to the left**	***per* → %** **number →** *c.* **[money]**

Notations

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

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

Mutual Information:

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

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

find random partion $P = \{P_1, P_2\}$ of $t_1, ..., t_m$
while (improving $I(P; Q)$) do
  find partion $Q = \{Q_1, Q_2\}$ of $x_1, ..., x_n$
    that maximizes $I(P; Q)$
  find partion $\{P_1, P_2\}$ of $t_1, ..., t_m$
    that maximizes $I(P; Q)$
end

"Flip-Flop" algorithm

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

Stopping criterion:

$I(P;Q)$ does not increases 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 occurence the sense 1;
if $x_i \in Q_2$ then assign the occurence 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}***, $Q_2 =$ ***{décision, 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;**
$c_1, c_2, ..., c_I$ **are the contexts of the ambiguous word in the corpus;**
$w_1, w_2, ..., w_J$ **are the words being used as disambiguating features.**

Parameters of the model (μ):

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

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

$$l(C \mid \mu) = \textbf{\textit{log}}\ \Pi_{i=1}^{I} P(c_i) = \textbf{\textit{log}}\ \Pi_{i=1}^{I} \Sigma_{k=1}^{K} P(c_i \mid s_k) P(s_k) =$$
$$\Sigma_{i=1}^{I} \textbf{\textit{log}}\ \Sigma_{k=1}^{K} P(c_i \mid s_k) P(s_k)$$

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

$$P(c_i \mid s_k) = \Pi_{w_j \textbf{ in } c_i} P(w_j \mid s_k).$$

Procedure:

1. **Initialize the parameters of the model μ randomly.**
2. **While $l(C \mid \mu)$ is improving, repeat**
   a. **E-step: estimate the (posterior) probability that the sense s_k generated the context c_i:**
   $$h_{ik} = \frac{P(c_i|s_k)}{\Sigma_{k=1}^{K} P(c_i|s_k)}$$
   b. **M-step: re-estimate the parameters $P(w_j \mid s_k)$, $P(s_k)$, by way of MLE:**
   $$P(w_j \mid s_k) = \frac{\Sigma_{\{c_i \mid w_j \text{ in } c_i\}} h_{ik}}{Z_k} \quad \text{and} \quad P(s_k) = \frac{\Sigma_{i=1}^{I} h_{ik}}{\Sigma_{k=1}^{K} \Sigma_{i=1}^{I} h_{ik}} = \frac{\Sigma_{i=1}^{I} h_{ik}}{I}$$
   where Z_k is a normalizing constant:
   $$Z_k = \Sigma_{k=1}^{K} \Sigma_{\{c_i \mid w_j \text{ in } c_i\}} h_{ik}.$$

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

comment: Initialization
for all senses s_k of w do
  F_k = the set of features (words) in s_k dictionary definition
  $E_k = \emptyset$
end
comment: one sense per discourse
while (at least one E_k changed in the last iteration) do
  for all senses s_k of w do
    comment: identify the contexts c_i bearing the sense s_k
    $E_k = \{c_i \mid \exists f_m \in F_k \ : \ f_m \in c_i\}$
  end
  for all senses s_k of w do
    comment: retain the features f_m which best express sense s_k
    $F_k = \{f_m \mid \forall n \neq k \ \frac{P(s_k|f_m)}{P(s_n|f_m)} > \alpha\}$ where $P(s_i|f_m) = \frac{C(f_m,s_i)}{\Sigma_j C(f_m,s_j)}$
  end
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 $s_k = \textit{argmax}_{s_i} P(s_i)$ where $P(s_i) = \frac{\Sigma_{m \in F_i} C(f_m, s_i)}{\Sigma_j \Sigma_{m \in F_i} C(f_m, s_j)}$
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 *ash*.**
1	0	**The *ash* 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 score(s_k) = overlap(D_k, $\cup_{v_j}$ in $_c E_{v_j}$)
4 end
5 comment: Disambiguation
6 choose $s' = \text{argmax}_{s_k}$ 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)
bass	**musical senses**	MUSIC
	fish	ANIMAL, INSECT
star	**space object**	UNIVERSE
	celebrity	ENTERTAINER
	star shaped object	INSIGNIA
interest	**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       score(s_k) = Σ_{w_j} in c δ(t(s_k), w_j)
1       comment: score = #words compatible with the category of s_k
5    end
6    comment: Disambiguation
7    choose s' = argmax_{s_k} 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

comment: categorize words/topics on contexts c
for all words w_j in the vocabulary do
  $C_j = \{c \mid w_j \text{ in } c\}$
end
for all topics t_l do
  $T_l = \{c \mid t_l \in t(c)\}$
end
for all words w_j and all topics t_l do
  $P(w_j \mid t_l) = |C_j \cap T_l| / \Sigma_j |C_j \cap T_l|$
end
for all topics t_l do
  $P(t_l) = (\Sigma_j |C_j \cap T_l|) / (\Sigma_l \Sigma_j |C_j \cap T_l|)$
end
comment: Disambiguation
for all senses s_k of w occurring in c do
  score(s_k) = log P$(t(s_k))$ + $\Sigma_{w_j \text{ in } c}$ log $P(w_j \mid t(s_k))$
choose $s' =$ argmax$_{s_k}$ 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.