

# Assignment 9 + Word Sense Disambiguation

## (SNLP Tutorial 10)

Vilém Zouhar, Awantee Deshpande, Julius Steuer

29th June, 1st July

# Assignment 9

- Exercise 1: Feature Engineering, Classification
- Bonus: Support Vector Machines

# Word Sense Disambiguation

Apple is full of vitamins.

Apple was struggling last quarter.

Apple was thrown away from the meeting.

# Word Sense Disambiguation

Apple is full of vitamins.

Apple was struggling last quarter.

Apple was thrown away from the meeting.



# Word Sense Disambiguation

Apple is full of vitamins.

Apple was struggling last quarter.

Apple was thrown away from the meeting.



$$f(w, C) = s \in S_w$$

$$f(\text{Apple}, * \text{ was thrown away from the meeting}) \in \{\text{fruit, company}\}$$

# Word Sense Disambiguation

Machine translation:

- Apfel ist voller Vitamine.
- Apple ist voller Vitamine.
- Apfel hatte im letzten Quartal Probleme.
- Apple hatte im letzten Quartal Probleme.

# Word Sense Disambiguation

Machine translation:

- Apfel ist voller Vitamine.
- Apple ist voller Vitamine.
- Apfel hatte im letzten Quartal Probleme.
- Apple hatte im letzten Quartal Probleme.

Information retrieval:

- Query: Apple vitamins
- Relevant document: benefits of eating apples

# Word Sense Disambiguation

Machine translation:

- Apfel ist voller Vitamine.
- Apple ist voller Vitamine.
- Apfel hatte im letzten Quartal Probleme.
- Apple hatte im letzten Quartal Probleme.

Information retrieval:

- Query: Apple vitamins
- Relevant document: benefits of eating apples

Dialogue systems

Spelling correction



# One sense per ...

One sense per discourse

- One meaning per word+document

# One sense per ...

One sense per discourse

- One meaning per word+document

One sense per collocation

- Nearby words help determine the sense

# Dictionary

- Dictionary/Thesaurus:  $\forall w, s \in S_w : D(s) = \text{description of sense } s$
- Context:  $\forall w, C(w) = \text{context of word } w \text{ in a specific occurrence}$

## Lesk's Algorithm

- Idea: Sense  $s_i$  of ambiguous word  $w$  is likely to be the correct sense if many of the words used in the dictionary definition of  $s_i$  are also used in the definitions of words in the ambiguous word's context.

$$s_{opt} = \underset{s_k}{\operatorname{argmax}} \operatorname{sim} \left( D(s_k), \bigcup_{v_j \in C} E(v_j) \right)$$

## Similarity

$$\frac{2|X \cap Y|}{|X| + |Y|}$$

$$\frac{2|X \cap Y|}{|X \cup Y|}$$

$$\frac{|X \cap Y|}{\sqrt{|X| \cdot |Y|}}$$

- Advantages? Disadvantages?

## Simplified Lesk's Algorithm Example

Sentence: The *bank* can guarantee deposits will eventually cover future tuition costs because it invests in adjustable-rate mortgage securities.

### Senses

- *bank*<sup>1</sup> Gloss/Def<sup>n</sup>: a financial institution that accepts deposits and channels the money into lending activities.  
e.g.: “She cashed a cheque at the bank”.
- *bank*<sup>2</sup> Gloss/Def<sup>n</sup>: sloping land (especially the slope beside a body of water).  
e.g.: “They had a picnic on the river bank”.

## Simplified Lesk's Algorithm Example

Sentence: The *bank* can guarantee deposits will eventually cover future tuition costs because it invests in adjustable-rate mortgage securities.

### Senses

- *bank*<sup>1</sup> Gloss/Def<sup>n</sup>: a financial institution that accepts deposits and channels the money into lending activities.  
e.g.: “She cashed a cheque at the bank”.
- *bank*<sup>2</sup> Gloss/Def<sup>n</sup>: sloping land (especially the slope beside a body of water).  
e.g.: “They had a picnic on the river bank”.

```
from nltk.corpus import senseval
hard, interest, line, serve = senseval.fileids()
line_instances = senseval.instances(line)
```

- Improvements include weighting by measures like IDF

# Supervised Disambiguation

## Bayes Decision

$$\begin{aligned}\hat{s} &= \arg \max_s p(s|C) = \arg \max_s \frac{p(C|s) \cdot (p(s))}{p(C)} \\ &= \arg \max_s p(C|s) \cdot (p(s))\end{aligned}$$

## Naïve Bayes

$$p(C|s) = \prod_{x \in C} p(x|s)$$

- Estimate by MLE counts (+ smoothing)
- Independence within context
- Position in context does not matter
- Advantages? Disadvantages?
- What kind of feature vectors can exist?

# Supervised Disambiguation Features Example

Sentence: Transactions on a deposit account of the *bank* are recorded in books, and the resulting balance is recorded as its liability.

- Collocational features:

$[w_{i-3}, POS_{i-3}, w_{i-2}, POS_{i-2}, \dots, w_{i+3}, POS_{i+3}]$

[account, NN, of, PP, the, DT, ... ]

$[w_{i-2}, w_{i-1}, w_{i+1}]$

- Bag of Word Features

Let  $V : \{\text{institution, account, water, land}\}$

Vector:  $[0, 1, 0, 0]$  for given sentence

# Flip-Flop Algorithm

- Machine translation is able to choose the right sense (assuming different senses have different translations)
- Apple was struggling last quarter.  
Apple hatte im letzten Quartal Probleme.
- Apple is full of vitamins.  
Apfel ist voller Vitamine.
- Translations (in German): {Apfel, Äpfel, Apple}
- Indicator words: {struggling, quarter, full, vitamins} (stopwords removed)



# Flip-Flop Algorithm

- Machine translation is able to choose the right sense (assuming different senses have different translations)
- Apple was struggling last quarter.  
Apple hatte im letzten Quartal Probleme.
- Apple is full of vitamins.  
Apfel ist voller Vitamine.
- Translations (in German): {Apfel, Äpfel, Apple}
- Indicator words: {struggling, quarter, full, vitamins} (stopwords removed)

Partition translated words ( $\{Q_1, Q_2\}$ ) and indicator words ( $\{P_1, P_2\}$ ) to maximize:

$$I(P; Q) = \sum_{i \in Q, t \in P} \log \frac{p(i, t)}{p(i) \cdot p(t)}$$

## Flip-Flop Algorithm

- 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  $x_1, \dots, x_n$  that maximises  $I(P;Q)$
    - 4   ▶ find partition  $P = \{P_1, P_2\}$  of  $t_1, \dots, t_m$  that maximises  $I(P;Q)$
  - 5 end
- $t_i$  : translations of the ambiguous word
  - $x_i$  : indicator words
  - $I(P;Q)$  monotonically increases until convergence

## Flip-Flop Algorithm

- 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  $x_1, \dots, x_n$  that maximises  $I(P;Q)$
- ④   ▶ find partition  $P = \{P_1, P_2\}$  of  $t_1, \dots, t_m$  that maximises  $I(P;Q)$
- ⑤ end

- $t_i$  : translations of the ambiguous word
- $x_i$  : indicator words
- $I(P;Q)$  monotonically increases until convergence

- Disambiguation

Determine  $x_i$

if  $x_i \in Q_1$  assign sense 1

if  $x_i \in Q_2$  assign sense 2

# Unsupervised Disambiguation (EM Algorithm)

- Idea: Random initialisation followed by parameter estimation
- Parameters?  $P(v_j|s_k)$  and  $P(s_k)$
- Maximise log-likelihood  $\log \prod_i \sum_k P(c_i|s_k)P(s_k)$
- E step:  $h_{ik} = \frac{P(c_i|s_k)P(s_k)}{\sum_l P(c_i|s_l)P(s_l)}$
- M step:  $P(v_j|s_k) = \frac{\sum_i C(v_j \in c_i) \cdot h_{ik}}{\sum_j \sum_i C(v_j \in c_i) \cdot h_{ik}}$   
 $P(s_k) = \frac{\sum_i h_{ik}}{\sum_k \sum_i h_{ik}}$
- Disambiguation:  $s_{opt} = \operatorname{argmax}_{s_k} [\log P(s_k) + \sum_{v_j \in C} \log P(v_j|s_k)]$

# Semi-Supervised Disambiguation (Yarowsky Algorithm)

- Utilises one sense per discourse and one sense per collocation
- Algorithm:

- 1 In a large corpus, identify all examples of a polysemous word, and store their contexts as an untagged training set.

e.g.

The company *plant* is still operational. . .

The region abounds in *plant* life. . .

The classification of *plant* and animal kingdoms. . .

- 2 For each sense of the word ( $s_1 \dots s_k$ ), identify collocations representative of the sense, and tag all the sentences from (1) which contain the seed collocation with the respective label.

e.g.

Sense 1: The company *plant* is still operational

Sense 2: The region abounds in *plant* life. . .

Sense 2: The classification of *plant* and animal kingdoms

# Yarowsky Algorithm

- 3
  - a) Train on the seed sets (Sense 1, Sense 2).
  - b) Apply the obtained classifier on the entire sample set. Only retain those tags that are above a certain probability threshold. Add these examples to the seed set.
  - c) Use one sense per discourse to augment and correct the available data.  
e.g.  
Sense 1: The company *plant* is still operational  
? → Sense 1: The *plant* was shut down due to inflation.
  - d) Repeat (3a) to (3c)
- 4 Hold training parameters constant, and the algorithm will converge on the residual set.
- 5 Apply the classifier to new data or original untagged data.

# Resources

- ① UdS SNLP Class, WSD: <https://teaching.lsv.uni-saarland.de/snlp/>
- ② Classical Statistical WSD: <https://www.aclweb.org/anthology/P91-1034.pdf>
- ③ WSD: <https://www.cs.toronto.edu/~frank/csc2501/Lectures/8%20Word%20sense%20disambiguation.pdf>
- ④ Lesk Algorithm: <https://www.c-sharpcorner.com/article/lesk-algorithm-in-python-to-remove-word-ambiguity/>
- ⑤ Yarowsky Algorithm: [https://www.coli.uni-saarland.de/courses/comsem-10/material/Victor\\_Santos\\_Yarowsky.pdf](https://www.coli.uni-saarland.de/courses/comsem-10/material/Victor_Santos_Yarowsky.pdf)
- ⑥ <https://www.aclweb.org/anthology/P95-1026.pdf>
- ⑦ <https://web.stanford.edu/~jurafsky/slp3/slides/Chapter18.wsd.pdf>