INF5830
Word Sense Disambiguation today

Group session 06/11/2017
Excercises with WSD

Andrey Kutuzov
andreku@ifi.uio.no

6 November 2017

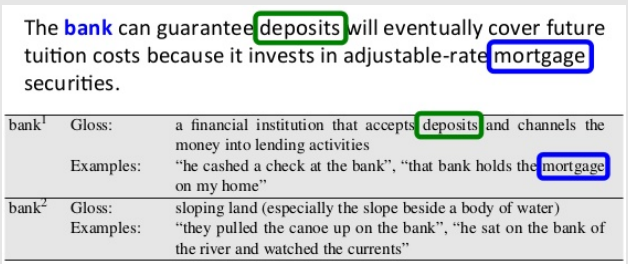
Word Sense Disambiguation

Preparation:

- Clone the git repository at
  `https://github.uio.no/andreku/INF5830`
- *pip3 install –upgrade nltk*
- *python3 -m nltk.downloader 'wordnet'*
- *python3 -m nltk.downloader 'punkt'*
- *python3 -m nltk.downloader 'averaged_perceptron_tagger'*
- *python3 -m nltk.downloader 'semcor'*

Word Sense Disambiguation

Lesk algorithm


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

bank[1]	Gloss:	a financial institution that accepts deposits and channels the money into lending activities
	Examples:	"he cashed a check at the bank", "that bank holds the mortgage on my home"
bank[2]	Gloss:	sloping land (especially the slope beside a body of water)
	Examples:	"they pulled the canoe up on the bank", "he sat on the bank of the river and watched the currents"

- The classical approach to WSD from the 80s:
  1. We need to disambiguate word *x* in the sentence *S*
  2. Compute lexical intersections *i* between words in *S* and the different dictionary definitions of *x* ('signatures' or 'glosses').
  3. Choose the definition with the highest *i* as the sense of *x* in *S*.

Let's implement it!

- Dictionary definitions can be taken from the Wordnet.
- Use NLTK+Wordnet to implement the lacking functions in *lesk2implement.py*
- Is your implementation better than the first sense baseline?

Word Sense Disambiguation

Moving on

- Later, many improvements were proposed to Lesk:
  - add manually annotated sentences to signatures,
  - weight words with TF/IDF...
  - etc.
- Many of them can be found in the PyWSD package:
  - *pip3 install pywsd*
  - *from pywsd.lesk import simple_lesk*
  - *answer = simple_lesk(sentence, 'bank', pos='n')*
  - or all-words disambiguation:
  - *from pywsd import disambiguate*
  - *output = disambiguate(sentence)*

Word Sense Disambiguation

Evaluation

- Let's evaluate these methods on the *SemCor* manually annotated corpus.
  - *from nltk.corpus import semcor*
  - *semcor.tagged_sents(tag='both')[10]*
- Lexical sample task:
  - disambiguate the word '*form*' in the *SemCor* with MFS, your Lesk implementation, and adapted_lesk from PyWSD
  - use the *semcor_lesk.py* script
- Is it difficult to beat the MFS baseline?
- Test on other nouns.
- Add Wordnet definitions of classes to the results reporting.
- Implement some way to do error analysis: print out several sentences classified incorrectly.

Word Sense Disambiguation

Supervised WSD

- We can do without any definitions or other knowledge sources.
- Just use machine learning.
- *pip3 install –upgrade scikit-learn*
- *pip3 install –upgrade matplotlib*
- Train a supervised model to disambiguate '*form*'.
- Use the *semcor_supervised.py* script (it employs SVM classifier).
- Is it better than Lesk-based methods?
- Try other classifiers (decision trees, logistic regression...) or try enriching the vocabulary with n-grams, etc

Word Sense Induction

Clustering word senses

- Let's go fully unsupervised.
- Infer the senses from the data!
- Add your code to *semcor_wsi.py* to cluster sentences.
- Compare the inferred clusters to *Wordnet* senses.
- How can we get rid of the necessity to state the desired number of clusters?

Using web services API to disambiguate

Optional

- Implement a system which gets WSD decisions from *Jobim* web service.
- Online demo: `http://ltbev.informatik.uni-hamburg.de/wsd`
- API description: `http://ltmaggie.informatik.uni-hamburg.de/jobimtext/wsd/#APIs`