Project Text Analysis – Assignment 4

Malvina Nissim, Lennart Kloppenburg
m.nissim@rug.nl, lennyklb@gmail.com

20 May 2015

The aim of this assignment is to get you acquainted with **disambiguating words**. You will use WordNet again, and the implementation of a word sense disambiguation algorithm which you can find in NLTK. This algorithm is called the Lesk algorithm (for a simple description of the Lesk algorithm you can actually refer to the WIkipedia page: `http://en.wikipedia.org/wiki/Lesk_algorithm`). The code that you will develop for this assignment will come useful in your final project, when you will have to disambiguate entities in order to assign to them the "correct" hypernym synset.

- use Nestor for submission
- **Deadline**: Tuesday 26th, 23:59
- hand in two files, one `.py` and one `.txt/.pdf`:
  - one python script that does what is asked in Exercise 1 (disambiguation of all ambiguous nouns in the chosen dataset)
  - a `.txt` or `.pdf` file (please no `.docx`) with all answers to the questions in Exercise 1 and any comments you might have.

Exercise 1 – Word Sense Disambiguation

Word Sense Disambiguation (WSD) is the task of determining the intended sense of an ambiguous word in a given context. For example, the word "mouse" in (a) "I must see the doctor because I'm having pain when using the mouse" refers to the computer appliance, whereas in (b) "The cat was chasing the mouse", it refers to the little grey animal. There exist rather sophisticated algorithms that deal with the resolution of sense ambiguity, but we are going to keep it simple, both intuitively as well as practically.

In NLTK, you can find and use the implementation of one of the simplest disambiguation algorithms, which is called "Lesk", and uses WordNet as a sense inventory. You already know how to use WordNet in NLTK! As a reminder, you can list all senses (= synsets) of a word like this, specifying the part-of-speech (for example "n" for noun):

```
>>> from nltk.corpus import wordnet
>>> for ss in wordnet.synsets("mouse", "n"):
...     print(ss, ss.definition())
Synset('mouse.n.01') any of numerous small rodents typically resembling ...
Synset('shiner.n.01') a swollen bruise caused by a blow to the eye
Synset('mouse.n.03') person who is quiet or timid
Synset('mouse.n.04') a hand-operated electronic device that controls the ...
```

(Note and remember: why is there 'shiner.n.01' among the senses of "mouse"?)

What you want, though, is a way of telling which of the available synsets for "mouse" is the appropriate one in Example (a) and in Example (b) above. And this is what the Lesk algorithm will try to do for you, after you fed it some text and the specific word you want to disambiguate. To work, it will exploit the surrounding context. Below you see how you can make it work:

```
>>> from nltk.wsd import lesk
>>> from nltk import word_tokenize
>>> sent = word_tokenize("I must see the doctor because I'm having pain
... when using the mouse.")
>>> word = "mouse"
>>> pos = "n"
>>> print(lesk(sent, word, pos))
Synset('mouse.n.04')
```

If you want to see everything together, of course you can just combine the code easily:

```
>>> from nltk.corpus import wordnet
>>> from nltk.wsd import lesk
>>> from nltk import word_tokenize
>>> sent = word_tokenize("I must see the doctor because I'm having pain
... when using the mouse.")
>>> word = "mouse"
>>> pos = "n"
>>> print(lesk(sent, word, pos))
>>> print("All possible senses:")
>>> for ss in wordnet.synsets(word, pos):
...     print(ss, ss.definition())
```

and this would the output:

```
Synset('mouse.n.04')
All possible senses:
Synset('mouse.n.01') any of numerous small rodents typically resembling ...
Synset('shiner.n.01') a swollen bruise caused by a blow to the eye
Synset('mouse.n.03') person who is quiet or timid
Synset('mouse.n.04') a hand-operated electronic device that controls the ...
```

This isn't magic, of course, it's a rather simple algorithm (though quite efficient in its simplicity) and prone to error:

```
>>> from nltk.corpus import wordnet
>>> from nltk.wsd import lesk
>>> from nltk import word_tokenize
>>> sent = word_tokenize("The cat chased the mouse, but didn't catch it.")
>>> word = "mouse"
>>> pos = "n"
>>> print(lesk(sent, word, pos))
>>> print("All possible senses:")
>>> for ss in wordnet.synsets(word, pos):
...     print(ss, ss.definition())
Synset('mouse.n.04')
All possible senses:
Synset('mouse.n.01') any of numerous small rodents typically resembling ...
Synset('shiner.n.01') a swollen bruise caused by a blow to the eye
Synset('mouse.n.03') person who is quiet or timid
Synset('mouse.n.04') a hand-operated electronic device that controls the ...
```

As you can see, the synset for "mouse.n.04" is returned, while it should have been "mouse.n.01". However, we can still use it, of course!

What you have to do

You will have to choose some data to work with, run the Lesk algorithms for all nouns that are ambiguous (how do you know if a noun is ambiguous?), and answer some simple questions, just to reflect on what you have done. Unless you do it manually, there is no way to say how accurate the algorithm is on your data because you do not have a gold standard for sense annotation for all nouns. But the main aim of this assignment isn't getting correct data (after all, you are just running an existing algorithm), rather to make you acquainted with ambiguity, and possible ways to tackle it for your final project. The code that you will write you will be able to reuse, of course. Also, reflecting on what happens might prompt some good ideas for dealing with this problem in the project, both in your data and on the Wikipedia pages.

Choose your data

You can pick one of the following three options:

1. work with all of your annotated files
2. work with the Wikipedia pages for which you found link when you did the annotation
3. work with BOTH sets of data

Select the words you will work with

You will be working with **nouns**. By now, you know very well how to get this information out of raw text. If you are using your annotated data, you already have a column that specify parts-of-speech, so you can obviously use it to deduce what the nouns are. In case

you are using Wikipedia data... what do you have to do? If a noun is monosemous (it has one sense/synset only), there is no need to disambiguate it. Thus, you will run the Lesk algorithm only on those nouns that are ambiguous — you will have to write code for this.

Disambiguate the selected words

Once you have identified polysemous (i.e ambiguous) words, you will run the Lesk algorithm for each of those. Think about the **context**. You can use as context the sentence in which the ambiguous words appear, or possibly more than one sentence, or possibly the whole text. This is up to you, and you are welcome to experiment with contexts of different sizes. Please, **report** on this in the document that you submit. For each of the words then you will have an assigned synset. As we said, unless you check it manually, you cannot know whether it is correct since you do not have a reference gold standard for this against which you can measure the performance of the algorithm.

Report on what you observe

Now you can write a few comments on what you observe by doing this (answering some of the questions would require extra-coding). Please, make sure that you mention which data you used. If you used BOTH Wikipedia and your annotated documents, you can separate out the answers and observe whether there are any differences across the two datasets.

1. what is the proportion of polysemous words per document?
2. did all documents contain at least one polysemous word?
3. what is the average number of senses for the polysemous words you found?
4. for each number of different senses found, please list the number of words that have them (for example: 3 words showed 5 senses, 10 words showed 4 senses, and so on)
5. pick five to ten cases randomly:
   - are they correct? if so, why do you think it works?
   - are they wrong? if so, why do you think it doesn't work?

   on this, you can also comment regarding the extension of the context you chose to use.
6. did it happen that the same word was assigned a different sense in the same text? do you think it can depend on the extension of the context you decide to use? if so, why?
7. does what you can observe from the above correspond to what you were expecting?
8. how do you think you could use disambiguation in your Wikification project? You can add a reflection on varying context size in case you tried it or just thinking about it.