

CS838-1 Advanced NLP Homework 1

Due 2/13/2007 in class

Instructor: Jerry Zhu, jerryzhu@cs.wisc.edu

Type your answers and hand in a printed version to the instructor in class on the due date. The homework is worth 50% if it is no later than 48 hours (you may email me a pdf file), and worth nothing after that. I will not accept homeworks to the TA or in the physical mailbox.

Note: This homework assumes you know basic unix commands. Please talk to the TA for linux/unix help. Alternatively, you can do everything under Windows (not recommended though).

1 Sentence Segmentation

Download the particular version of *Alice's Adventures in Wonderland* from <http://www.cs.wisc.edu/~cs838-1/dataset/alice/alice.txt>. This is the document we'll be working on.

Spend 15 minutes to write your own sentence boundary detector, in any program language you like. It should put all words of a sentence on a single line, even if in the original text they are on multiple lines. On the other hand, put different sentences on different lines. (In short, one sentence per line.) Your program can be as simple as detecting the usual sentence boundaries: `. ? !`. Apply it to *Alice* and inspect the output.

Then, download MXTERMINATOR, a mature sentence boundary detector, from <http://home.comcast.net/~adwaitr/jmx.tar.gz>. Follow the instruction in `MXTERMINATOR.html`. If you use *tsh*, simply do `setenv CLASSPATH mxpost.jar` then you should be able to run it. Use the `eos.project` that comes with the package. Apply it to *Alice*.

2 Tokenization

Once you have segmented out sentences, it's time to separate individual words. Download the Penn Treebank tokenizer from <http://www.cis.upenn.edu/~treebank/tokenization.html>. This is a UNIX *sed* program. Run it with `sed -f`. It needs an input file with one sentence per line. Apply the tokenizer to the processed *Alice* corpus.

3 Stemming

Download the Porter stemmer in the language you like from <http://www.tartarus.org/~martin/PorterStemmer/>. Run the stemmer on *Alice* from the previous step. You will notice that it maps all words to lower case, and some words look funny.

Questions (100 points total)

1. (10 points) Briefly tell us your sentence boundary detector design. How many sentences do you get?
2. (10 points) Do you observe differences in sentence boundaries your program produced vs. MXTERMINATOR? If so, give two examples.
3. (10 points) Observe Treebank tokenizer's output. Do you see things it does wrong? Give two examples.
4. (10 points) List 5 words that the Porter Stemmer did not handle properly (show the stemmed version too).
5. (10 points) Do not strip punctuations or otherwise change the tokens out of the stemmer. How many *word tokens* and *word types* are there?
6. (10 points) List the top 10 most frequent words (they can be punctuations) and their counts.
7. (10 points) Using any plotting software, plot rank (x -axis) vs. count (y -axis) for all words. Each word would be a dot in such a plot. In a second plot, plot the same thing but use log scale on both axes. Discuss whether your plot fits Zipf's law.
8. (20 points) Please find a partner for this (and only this) question. Find a small corpus (e.g., a story, a book, anywhere between 100KB to a few MB in size) in a language *other than English*. For examples see the ACL wiki <http://aclweb.org/aclwiki/index.php?title=Corpora>. Try to produce a *rank vs. count plot* for the corpus. Note the above software may no longer work on your language. *Briefly discuss the special properties of the language* that are important for this question, and *how you processed it*.
9. (10 points) Let us assume the Miller's monkey has only two keys: A and white space, and it hits them with probability p and $1 - p$ respectively. Derive the rank and frequency function relation on monkey words.