BM1 Advanced Natural Language Processing
University of Potsdam, Winter 2016/17

Assignment 1: Modelling Language

Tatjana Scheffler, Ph.D.

Due: Friday, November 4, 10:00 a.m.

Use Python 2 and the Natural Language Toolkit (NLTK) for your exercises. The NLTK book ("Natural Language Processing with Python") is available online at `http://www.nltk.org/book_1ed/` You should also familiarize yourselves with working with unicode and UTF-8 in Python. The session on Tuesday, October 25 can be used to get up to speed with Python, NLTK, and the packages used in the assignments below.

Problem 1: Zipf's Law

Empirically verify Zipf's law. Use the following freely available corpora:

- King James Bible,
  http://www.gutenberg.org/cache/epub/10/pg10.txt
- The Jungle Book,
  http://www.gutenberg.org/cache/epub/35997/pg35997.txt
- SETIMES Turkish-Bulgarian parallel newspaper text,
  http://opus.lingfil.uu.se/download.php?f=SETIMES2/bg-tr.txt.zip

For each corpus, give a list of unique word forms sorted by descending frequency. Preferably using the Python libraries `numpy` and `matplotlib`, plot the frequency curves for the corpora. Make sure to provide both a linear curve, and a log-log curve (see methods `matplotlib.pyplot.plot` and `matplotlib.pyplot.loglog`).

Provide a brief discussion of the findings, as well as the source code.

Problem 2: Random Text Generation

In this assignment, you will reimplement the "Dissociated Press" system that was developed by MIT students in the 1970s (see Wikipedia). The purpose of this system is to generate random text from an n-gram model over a corpus.

Train an `nltk.model.ngram.NgramModel` using a corpus of your choice (from Problem 1 or elsewhere), and name it `ngram`. You can then use `ngram[context]` to determine the probability distribution for the next word given the previous

$n - 1$ words. Given this distribution, you can use the method `generate` from the NLTK class `ProbDistI` to generate the next random word.

Use your system to produce a number of text samples, 100 words in length per each. Vary n from 2 to 4. Submit a few interesting texts that your system generates, and discuss how the quality (and creativity) of the generated outputs changes with n. Also submit your source code, and document any dependencies, such as links to the selected corpora.

Problem 3: Pointwise Mutual Information

In statistical NLP we frequently make independence assumptions about relevant events which are not actually correct in reality. We are asking you to test the independence assumptions of unigram language models.

Pointwise mutual information,

$$\text{pmi} = \frac{P(X_t = w_1, X_{t+1} = w_2)}{P(X_t = w_1)P(X_{t+1} = w_2)} \approx \frac{f(w_1, w_2)N}{f(w_1)f(w_2)}$$

is a measure of statistical dependence of the events $X_t = w_1$ and $X_{t+1} = w_2$; $f(w)$ is the absolute frequency of word w and N is the total size of the corpus. If the probability of the next word in the corpus being w_2 is unaffected by the probability of the previous word being w_1, then $\text{pmi}(w_1, w_2) = 1$; otherwise the pmi is higher or lower than one.

Calculate the pmi for all successive pairs (w_1, w_2) of words in the King James Bible corpus. Words (not word pairs!) that occur in the corpus less than 10 times should be ignored. List the 20 word pairs with the highest pmi value and the 20 word pairs with the lowest pmi value.

Document and submit your observations and code. Discuss the validity of the independence assumption for unigram models.

Submit your solutions and code via email to `johannsmeier@uni-potsdam.de`