# Exercise Sheet 6

## Language Models and Text Processing

*Submit your solutions until **Monday, 18.04.2016, 12h00** by uploading them to ILIAS. Later submissions won't be considered. Every solution should contain the **name(s), email adress(es)** and **registration number(s)** of its (co-)editor(s).*

## 1 N-Grams (25 points)

Language models are composed of N-grams likelihoods (or probabilities), viz., of the probabilities of the sequences words of length $n$ a corpus contains.

In what follows, we will be considering the following corpus of 4 sentences:

| | |
|---|---|
| 1. | Chocolate is a kind of sweet made from cocoa beans. |
| 2. | Cocoa originated in ancient Mexico and is now grown around the globe. |
| 3. | Cocoa contains an active compound that makes our body produce oxytocin. |

Table 1: "Chocolate" corpus.

### 1.1 Bigrams and Unigrams (11 points)

Use the unigram and bigram formulas from the slides to compute

1. a unigram language model
2. a bigram language model

for the "Chocolate" corpus in Table 1.

### 1.2 Trigrams (11 points)

Write down the equation for trigram probability estimation, by modifying the following equation for bigrams

$$P(w_n|w_{n-1}) = \frac{C(w_n, w_{n-1})}{C(w_{n-1})}$$

Apply it to our "Chocolate" corpus to compute a trigram language model.

### 1.3 Chain Rule (4 points)

In probability theory, the so-called **chain rule**

$$\begin{aligned} P(x_n, \ldots, x_1) &= P(x_n|x_{n-1}, \ldots, x_1) \times P(x_{n-2}|x_{n-3}, \ldots, x_1) \times \cdots \times P(x_1) \\ &= \prod_{i=1}^{n} P(x_i|x_{i-1}, \ldots, x_1) \end{aligned}$$

is typically used to compute the (joint) probability of a **sequence** $x_1, \ldots, x_n$.

1. Can we use the chain rule to estimate the probability of a sentence in a corpus?
2. What is the main disadvantage of using the chain rule to estimate the probability of a sentence?
3. How does the chain rule relate to language (unigram, bigram and trigram) models?
4. Would the use of log-likelihoods and -probabilities[1] make the computation faster? If so, explain why.

## 2 Bonus: Java Language Generator (5 points)

As input, you specify the starting term of the sentence and its desired length. The result is a sentence of the given length where each word forms the most probable bigram with its previous word regarding the language model. The basic idea is that we create a language model. Then, we choose the most probable bigram according to the language model that starts with the given starting term. Then, we take the second term of that bigram, and search for the most probable bigram starting with it. We repeat that until the sentence length is reached.

Download the template `SentenceGenerator.java` and write the missing code. (You need to create a new project and import the LingPipe jar.)

Follow the following steps:

1. Look at the main() method. Add the path to the background text directory and create a language model. The language model is created as we have seen last week. Run it and have a look at the according method.
2. Go back to the main() method and comment in 2. FIND MOST PROBABLE SUCCESSOR OF A TOKEN. In this step, you have to implement the method that returns the most prob- able successor token for a term. Complete the according function following the instructions in the code template. Run it.
3. Go back to the main() method and comment in 3. GENERATE SENTENCE. In this step, you generate the sentence. Complete the according function following the instructions in the code template. Run it.

[1] The *log-probability* of a probability $p$ is $\log p$, and *log-likelihood* is the logarithm $\log \hat{p}$ of its MLE $\hat{p}$.