# Statistical Machine Translation

**Homework 3**

To be sent (pdf, Zip) to `waszczuk@phil.hhu.de` by 04.12.2018.

## Exercise 1 - Practice

This exercise directly follows the one from Homework 2. You can use your own solution or the one provided in the code for Homework 3. Download the archive (Eclipse project as Zip file) and save it. Start Eclipse and import the zip file as existing project. At the end of the session, you can export your project as zip archive and keep a copy of it (email, USB).

This exercise consists of several small tasks, which are easier to carry out in the given order. The Main class is `de.hhu.phil.smt.lm.Uebung3`. It already contains a lot of code. The output that this code produces should make sense after you solved the exercises.

### 1a

The method `score` calculates the probablility that the (Bigram-)model `de.hhu.phil.smt.lm.BigramModel` assigns to a *sentence*. Your first task is to implement a method `scoreFile`, which reads a file with test sentences and calculates their probability, assuming that the individual sentences are mutually independent.

Please use `data/train.de.tok.50.t50` (50 sentences) as test data. For training you can use as previously `data/train.de.tok.50.h1000` (1000 sentences), or `data/train.de.tok.50.h500000` with 500.000 training sentences.

### 1b

Most likely, you encountered a problem while doing exercise (1a): The probabilities get so small by multiplying that the computer can no longer represent them. Therefore, in practice, **log-probabilities** (log for logarithm) are used instead of the „bare "probabilities $p = \exp(\log p)$). So, for example,

$$\log \hat{P}Add1(w_i \mid w_{i-1}) = \log \frac{C(w_{i-1}w_i) + 1}{C(w_{i-1}) + |V|}$$

Usually the natural logarithm with base $e$ is used. The base of the logarithm is actually not important as long as it is used consistently.

Be careful: when dealing with log-probabilities, the following calculation rule is essential:

$$\log(a \cdot b) = \log a + \log b$$

This means that when we multiply probabilities, we need to add log-probabilities.

Implement the method `logScore`, which returns the log-probability of a model for a test sentence, and similarly `logScoreFile` for a test corpus. Java provides a method to calculate natural logarithms: `Math.log(double d)`.

## 2

For **Trigram-models** we assume that:

$$P(w_i \mid w_1, \ldots, w_{i-1}) = P(w_i \mid w_{i-2}, w_{i-1})$$

The maximum likelihood estimation of the parameters with Add-One Smoothing is accordingly

$$\hat{P}(w_i \mid w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}w_{i-1}w_i) + 1}{C(w_{i-2}w_{i-1}) + |V|}$$

Implement a **Trigram-model** with Add-One Smoothing as for bigram-model. Some parts are already written in `de.hhu.smt.lm.Trigram Model`. The missing parts are marked with `TODO`. [1] Treat unfamiliar words like in Homework 2.

## 3

Evaluation of the quality of the model. As a measure for the evaluation of language models, we often use **perplexity** of the language model in terms of test data $\boldsymbol{w}$ of length $n$. This is defined as

$$\begin{aligned} PP_{LM,\boldsymbol{w}} &= P(w_1 \ldots w_n)^{-1/n} \\ &= \exp[-\frac{1}{n}\sum_{i=1}^{n} \log P(w_i|h)] \end{aligned}$$

where $P$ is the probability which the language model assigns to the sequence $\boldsymbol{w}$. $h$ (*history*) represents the (N-1) previous words of $w_i$ depending on the N-gram model, e.g. $w_{i-1}$ in a bigram model. The smaller the perplexity, the better the language model.[2]

Implement a method to calculate the perplexity of a language model in function of a text corpus: `de.hhu.phil.smt.SMT.perplexity()`. The challenge here is that perplexity is defined in terms of a single sequence $\boldsymbol{w}$, while our test corpus is clearly a set of such sequences. This can be overcome by:

[1]You can also write the trigram language model from scratch. But your model has to implement, as the bigram model, the interface `de.hhu.phil.smt.lm.LanguageModel`.

[2]Also, by definition of perplexity, the smaller it is, the higher the probability of $\boldsymbol{w}$.

(a) Considering the test corpus as a single sequence of words consisting of concatenated test sentences.

(b) Assuming that the individual sentences in this concatenated representation are mutually independent.

The perplexity of the bigram and the trigram model on the test corpus is calculated (based on `de.hhu.phil.smt.SMT.perplexity()`) in the Main class. Are you surprised by the result?

**General note:** You do not have to stick exactly to the proposed structure of the code. If you are not sure, feel free to ask, but in any case, make sure that anyone can understand your code, for example by using comments. If the original entry point of the program no longer works, please add a readme file that specifies how the code should be compiled and executed.

## Exercise 2 - Theory

From Homework 2: we estimate bigram probabilities with add-one smoothing with

$$\hat{P}_{Add1}(w_i \mid w_{i-1}) = \frac{C(w_{i-1}w_i) + 1}{C(w_{i-1}) + |V|}$$

Show that this is an interpolation (weighted sum) of the maximum likelihood estimation and the uniform distribution estimation (each word in $V$ is equally probable), so that

$$\hat{P}_{Add1}(w_i \mid w_{i-1}) = \lambda_1 \hat{P}_{ML}(w_i \mid w_{i-1}) + \lambda_2 \frac{1}{|V|}$$

with $\lambda_1 + \lambda_2 = 1$.