# Statistical Natural Language Processing (SS-2018)
# Exercise 5 *

**Submission Deadline:** 25.05.2018, 23:59

## 1 Interpolation

Consider the following models for bigram and trigram probabilities:

$$P_2^I(w_2|w_1) = \lambda_1 \; P_1(w_2) + \lambda_2 \; P_2(w_2|w_1) \tag{1}$$

$$P_3^I(w_3|w_1, w_2) = \lambda_1 \; P_1(w_3) + \lambda_2 \; P_2(w_3|w_2) + \lambda_3 \; P_3(w_3|w_1, w_2) \tag{2}$$

(0.5 points) Preprocess the text in `data/train.txt` (tokenize and remove punctuation) and compute the unsmoothed unigram, bigram and trigram probabilities $P_1$, $P_2$ and $P_3$ for each n-gram in the text, $n \in \{1, 2, 3\}$. You may prune your model and remove trigrams and bigrams that appear once.

(0.5 points) Create a function that returns perplexity given relative frequencies of the test n-grams and training n-grams distribution.

(1,5 points) Compute the perplexity of the text in `data/test.txt`. First you should use smoothed trigram probabilities with Laplace Smoothing and $\alpha = 0.1$ as the trainings data. Then compute the perplexity using $P_3^I$ and $\lambda_1 = \lambda_2 = \lambda_3 = \frac{1}{3}$. Use your $P_n$ values trained during the first task for both language models. Compare your results and report for both models the 10 trigrams with highest probabilities from the training data. Do the trigrams and their probabilities differ or are they similar?

(0.5 points) What is the advantage of including lower-order n-grams? What are the disadvantages? Argue in terms of counts and explanatory power.

## 2 Cross Validation

Now we want to optimize our $\lambda$ values using cross validation. To reduce computational workload, use bigrams and the bigram model $P_2^I$. In the lecture you learned about leaving-one out cross validation. Instead of leaving only one word out for cross validation, you will implement a function where you always leave out a different fraction of the data for validation. This method is called K-fold cross validation, where K stands for the number of fractions (or folds as they are called). Split the data into K parts. Use each of the K parts once for validation and merge the rest for training. Calculate perplexity on the validation set. Take the average of the K perplexity values and return it. Use the text in `data/kfold.txt` for this task.

(1 point) Implement a 5-fold cross-validation function that takes $d$ as an input. The function should return the average of 5 perplexity values.

(1.5 points) Find the optimal values for $\lambda_1$ and $\lambda_2$ such that $\lambda_1 + \lambda_2 = 1$ and the perplexity of the cross validation is minimized. Use 2 digit floating point precision for the $\lambda$ values, e.g. $\lambda_1 = 0.65$, $\lambda_2 = 0.35$ and report the $\lambda$ values and how much your results improved. Plot the cross-validation perplexity for your tested $\lambda_1$s. Don't forget to label the axis.

*You can find an overview of the smoothing techniques used for this sheet (and more) with examples and descriptions in: `https://nlp.stanford.edu/~wcmac/papers/20050421-smoothing-tutorial.pdf` and `https://www.cs.jhu.edu/~jason/665/PDFSlides/lect05-smoothing.pdf` (with examples for 5-fold cross validation)

## 3 Witten-Bell Smoothing

Instead of using fixed $\lambda$ values for every n-gram, we can condition the $\lambda$ values on history. With this, we define Recursive Interpolation as:

$$\begin{aligned} P_n^I(w_i|w_{i-n+1},...,w_{i-1}) = \lambda_{w_{i-n+1},...,w_{i-1}}\ P_n(w_i|w_{i-n+1},...,w_{i-1}) \\ +\ (1-\lambda_{w_{i-n+1},...,w_{i-1}})\ P_{n-1}^I(w_i|w_{i-n+2},...,w_{i-1}) \end{aligned} \tag{3}$$

For Witten-Bell Smoothing, the Lambda parameters are defined as:

$$(1-\lambda_{w_{i-n+1},...,w_{i-1}}) = \frac{N_{1+}(w_1,...,w_{n-1})}{N_{1+}(w_1,...,w_{n-1}) + \sum_{w_n} count(w_1,...,w_{n-1},w_n)} \tag{4}$$

where $N_{1+}$ is the number of possible extensions of a history $w_I,...,w_{n-1}$ in training data:

$$N_{1+}(w_1,...,w_{n-1}) = |\{w_n : count(w_I,...,w_{n-1},w_n) > 0\}| \tag{5}$$

(1.5 points) Think of a corpus that contains the two words *grateful* and *linear* 1000 times each. However, *grateful* is followed by 12 different words while *linear* is followed by 230 different words due to the expressions *grateful for/that/because*. Thus, we are more likely to see an unknown bigram that starts with *linear* than *grateful*.

Using the definitions for Witten-Bell Smoothing, compute the Lambda parameters for $(1-\lambda_{grateful})$ and $(1-\lambda_{linear})$. What impact do these parameters have on the probabilities $P_n^I(\text{“}grateful\text{“}|h)$ and $P_n^I(\text{“}linear\text{“}|h)$ for some history $h$ and how will these probabilities presumable differ from each other?

## 4 Discounting

Consider the following discounting methods:

$$P_{disc}(w_3|w_1,w_2) = \frac{count(w_1,w_2,w_3)}{count(w_1w_2) + N_{1+}(w_1w_2)} \tag{6}$$

$$P_{disc}(w_2|w_1) = \frac{count(w_1,w_2)}{count(w_1) + N_{1+}(w_1)} \tag{7}$$

$$P_{disc}(w_1) = \frac{count(w_1)}{N+V} \tag{8}$$

where $N$ is the number of tokens and $V$ is the vocabulary size in the training corpus.

(0.5 points) These methods will discount some probabilities more than others. When is $P_{disc}(w_3|w_1w_2)$ very close to the naive estimate $c(w_1w_2w_3)/c(w_1w_2)$? When is it far less, i.e. heavily discounted?

(0.5 points) What if we changed the discounting formulas to exclude $N_{1+}$ and $V$? What would happen to the discounted estimates?

(2 points)

$$\hat{P}(w_3|w_1,w_2) = \begin{cases} P_{disc}(w_3|w_1,w_2) & \text{if } count(w_1,w_2,w_3) > 0 \\ \beta(w_1,w_2)\ \hat{P}(w_3|w_2) & \text{otherwise} \end{cases} \tag{9}$$

Using the above definition, you should construct $\beta(w_1,w_2)$ such that $\sum_{w_3} \hat{P}(w_3|w_1,w_2) = 1$. We know that $\sum_{w_2} \hat{P}(w_2|w_1) = 1$. (Hint: You may want to start by dividing the sum over $w_3$ into two sums).

## Submission Instructions

- You must form groups of 2 to 3 people
- Submit only 1 archive file in the ZIP format with name containing the MN of all the team members, e.g.:

  `Exercise_05_MatriculationNumber1_MatriculationNumber2.zip`

- Provide in the archive:

  i. your code, accompanied with sufficient comments
  ii. a PDF report with answers, solutions, plots and brief instructions on executing your code
  iii. a README file with the group member names, matriculation numbers and emails
  iv. Data necessary to reproduce your results

- The subject of your submission mail **must** contain the string [SNLP] (including the braces) and explicitly denoting that it is an exercise submission, e.g:

  `[SNLP] Exercise# Submission MatriculationNumber1 MatriculationNumber2`

- Depending on your tutorial group, please send your assignment to the **corresponding tutor**:
  - Mo. 16-18: Lukas Lange *s9lslang@stud.uni-saarland.de*
  - Th. 14-16: Harshita Jhavar *snlp18.thursday@gmail.com*
  - Fr. 10-12: Marimuthu Kalimuthu *mkalimuthu@lsv.uni-saarland.de*