1 Theoretical Review - Exercise 6

a) A researcher trained a language model \( m \) on a text \( T \) with and published a cross entropy of \( H_1(P_T, m) = 5.288 \). Another researcher used the same model and text of the first one but computed a value of \( H_2(P_T, m) = 7.629 \) and also a perplexity of \( PP(P_T, m) = 198 \). How do you explain the statements? What is every value measuring?

b) In the previous exercise we explored Laplace law smoothing (add-1), which is a special case of Lidstone law smoothing (add-\( \epsilon \)) for \( \epsilon = 1 \). In the case of ngram models, Lidstone law gives rise to the Floor discounting \( P_{FD} = \frac{N(w,h)+\epsilon}{N(h)+\epsilon V} \) mentioned in the class, where \( V \) is the size of the vocabulary (i.e. number of distinct words).

Show that \( P_{FD} \) is a linear interpolation between the Maximum Likelihood estimate for conditional probability \( P_{ML}(w|h) \) and a uniform distribution over all words in the vocabulary \( U(w) \).

Hint: The linear interpolation between \( f, g \) has the formula \( \alpha f + (1-\alpha) g \), where \( 0 \leq \alpha \leq 1 \).

c) In this exercise you decided that Absolute Discounting is discounting low counts improportionally much, so you decided to experiment with Linear Discounting. The discounting parameter is \( c(h) \) and the resulting distribution is

\[
P_{LD}(w|h) = \begin{cases} 
\frac{(1-c(h))N(w,h)}{N(h)} + \alpha(h)\beta(w|h) & \text{if } N(w,h) > 0 \\
\alpha(h)\beta(w|h) & \text{otherwise.}
\end{cases}
\]

Compute \( \alpha(h) \) so that \( P_{LD}(w|h) \) is appropriately normalised\(^1\). What property of \( \alpha(h) \) do you notice that the one of Absolute Discounting lacks? Do you notice any relation to another smoothing scheme already studied in the lecture?

d) As part of this exercise we also expect you to briefly review the practical part. If you decide to use the provided framework, try to familiarise yourself with the structure of the source code. Otherwise think briefly on the practical part and make a sketch of how you would implement it on your own.

You do not have to submit anything for this sub-task.

2 Practical Part - Exercise 7

This time, instead of ignoring missing entries, you will have to use smoothing appropriately to avoid problems in the calculation of perplexities. As a rule of thumb, you should extend your models with zero counts for each entry in your respective testing corpus and subsequently apply smoothing.

Also note that for this exercise we provided a framework\(^2\) which you may feel free to use completely or in part.

1. Tokenise the included file \( pg1344.txt \) and partition the contained words in 3 parts so that each contains roughly 60%, 20% and 20% of the words in the original. These parts will serve as your training \( T_{TRAIN} \), validation \( T_{VAL} \) and test sets \( T_{TEST} \) respectively.

2. Train\(^3\) unigram, bigram and trigram models \( m^1 \), \( m^2 \) and \( m^3 \) on \( T_{TRAIN} \). Complete/implement the smoothing algorithms of add-\( \epsilon \) and Absolute Discounting.

---

\(^1\)i.e.: make sure that \( \sum_w P_{LD}(w|h) = 1 \), in a way similar to Slide 18 of Chapter 5.

\(^2\)You are given two python3 files. One named Common06.py with generic tools. Another file is Ex06_framework.py, which contains a partial implementation of your task, including the bonus. Of course, the interesting parts are missing.

\(^3\)This part is already implemented.
3. Smoothen\(^4\) the n-gram models \(m^i\) to create \(\tilde{m}_p^i\) by tuning\(^5\) the single parameter \(p\) of each model. Your goal is to minimize the perplexity of the smoothened model w.r.t. \(T_{VAL}\)

\[
p^* = \arg\min_p PP(T_{VAL}, \tilde{m}_p^i).
\]

You should smoothen with the following scheme:

- Use add-\(\epsilon\) smoothing on \(m^1\) to get the smoothened model \(\tilde{m}_\epsilon^1\). Here the parameter to tune is \(\epsilon\).
- Use Absolute Discounting on \(m^2\) to get the smoothened \(\tilde{m}_\epsilon^2\). For a backing-off distribution use \(\tilde{m}_\epsilon^2\).
- Use Absolute Discounting on \(m^3\) to get the smoothened \(\tilde{m}_\epsilon^3\). For a backing-off distribution use \(\tilde{m}_\epsilon^3\).

If you need to specify boundaries, for the add-\(\epsilon\) optimum you may search for \(\epsilon \in [0.1, 5]\); for the Absolute Discounting you should look in \(d \in [0.01, 0.99]\). Why do the intervals have this form?

4. Report the best parameters \(\epsilon^*, d_{\epsilon}^2, d_{\epsilon}^3\) that you discovered and the corresponding perplexities \(PP(T_{VAL}, \tilde{m}_\epsilon^1), PP(T_{VAL}, \tilde{m}_\epsilon^2), PP(T_{VAL}, \tilde{m}_\epsilon^3)\). What do you notice?

5. We now want to evaluate our models. To do that, use \(T_{TEST}\) and report the perplexities \(PP(T_{TEST}, \tilde{m}_\epsilon^1), PP(T_{TEST}, \tilde{m}_\epsilon^2), PP(T_{TEST}, \tilde{m}_\epsilon^3)\).

6. Once we tune the parameters, we may further utilise the validation set. For the same parameters train new models \(\tilde{m}_\epsilon^1\) on the concatenation of the sequences \(T_{TRAIN}, T_{VAL}\). Report the perplexities \(PP(T_{TEST}, \tilde{m}_\epsilon^1), PP(T_{TEST}, \tilde{m}_\epsilon^2), PP(T_{TEST}, \tilde{m}_\epsilon^3)\).

**Bonus:** Implement/complete a method to compute the Kneser-Ney backing-off distribution \(\beta_{KN}\) resulting from the marginal constrain\(^6\).

Use your code to compute a distribution \(\beta_{KN}(w|h)\) based on the trigram counts of \(T_{TRAIN}\). Apply Absolute Discounting on \(m^3\) to create \(\tilde{m}_KN^3\) and use \(\beta_{KN}(w|h)\) as a backing-off distribution. Report again the best \(d_{KN}^3\) and the perplexities \(PP(T_{TRAIN}, \tilde{m}_{KN}^3), PP(T_{TEST}, \tilde{m}_{KN}^3)\). How does this model compare to the others? What is the intuition\(^7\) behind this variant of KN?

### 3 Submissions

This exercise sheet contains both exercises 6 and 7.

The theoretical part comprises Exercise 6. Please submit the theoretical part by mail to:

- s9tsback@stud.uni-saarland.de (if you are attending the Monday tutorial) by Fri Jun 13, 23:59 or
- kalofoli@ceid.upatras.gr (if you are attending the Tuesday tutorial) by Sun Jun 15, 23:59.

The practical part comprises Exercise 7. Please submit the practical part by mail to:

- s9tsback@stud.uni-saarland.de (if you are attending the Monday tutorial) by Fri Jun 20, 23:59 or
- kalofoli@ceid.upatras.gr (if you are attending the Tuesday tutorial) by Sun Jun 22, 23:59.

**Guidelines:** You have to demonstrate an effort to interpret the results of your source code; if you fail to produce reasonable results, you may still comment on what you expected to see.

Also please make sure that:

- you have included all files necessary to reproduce your results in one zipped folder, mentioning the MNs of every team member in your group, preferably in the format:
  
  Ex06-MemberName1(MN)-MemberName2(MN)-MemberName3(MN).zip

Submit files not requiring any modification or tweaking on our side to make them work, apart maybe from simple command line options.

- you have also included a brief report in pdf format answering the questions in this exercise and including your results.
- make clear the intentions\(^8\) of your code.
- make clear what question you are answering each time.

---

\(^4\)Remember: Depending on your implementation, before smoothing you may need to add zero entries to your models for all the unseen n-grams in \(T_{VAL}\), you should later do the same for \(T_{TEST}\).

\(^5\)For the tuning you may use any algorithm that searches for the minimum of a function, such as `numpy.optimization.fminsearch`. This is also implemented.

\(^6\)You may find more information on Slide 47 of Chapter 5.

\(^7\)i.e.: try to describe in words the event for which \(\beta_{KN}(w|h)\) attempts to assign a probability.

\(^8\)Try to use adequately descriptive variable/function names and to stick to your conventions. Python is a rather expressive language, but you may also use brief comments, if you deem appropriate or use another language.