COMPGI19 Assignment 1 - Language Models

Sebastian Riedel
Handed out: Friday 16th October
Due: Friday 6th November

1 Introduction

Language models are a fundamental tool of NLP and at the heart of applications such as speech recognition, handwriting recognition, and machine translation. They are particular useful in Noisy Channel approaches to these tasks, where the language model serves as a prior on text that generated the output we observe. In this assignment, you will design and implement your own specific language models and apply them on the provided rap corpus.

2 Rules and required Format

Your submission needs to consist of a report, in which you will summarise what you did and how you addressed the posed questions, and the code you hacked (filled in code stubs). The code you send needs to be compilable, as we will execute it on a hidden test set.

To get a feeling for "low-level" NLP implementation, for this assignment you are are not allowed to use ANY existing NLP packages, outside of the code provided and the `stat-nlp-book` code. The general UCL plagiarism rules apply.

2.1 Report

You will use the NIPS[1] style files for writing your reports. You can find out more about these styles on `https://nips.cc/Conferences/2015/PaperInformation/StyleFiles`. There are LaTeX, Word and RTF templates available. Your submission should be 6-8 pages maximum, with maximum 2 additional pages for references. Rules written here trump all the non-formatting instructions in the NIPS style files. If something is not clear regarding the style and instructions for the report, please post your question to Moodle. You need to submit a camera-ready version (not anonymised!) in PDF format, strictly.

2.2 Submission

Please upload your submission to Moodle as a zip file containing your report in PDF format, and your code in a zip file.

3 Problems

Problem 1. Tokenization (10pts) We have provided a text file with tokenized rap lyrics. Your task is to develop a regular expression based tokenizer that reproduces this tokenization. Start with the code in `object TokenizationQuestion` and improve it until your tokenization matches the provided gold tokenization. In your report, describe the regular expression you used. In case you have remaining mismatches, describe what needs to be done next.

Problem 2. Bar-Aware Language Model (total 55pts) In music, a *bar* (or measure) is a segment of time corresponding to a specific number of beats. The structure of a musical piece can be seen as a sequence of bars. The lyrics are often selected to match this structure, i.e. to go along with the beat. In the data, the opening of a bar is represented by the token `[BAR]` and the closing of a bar by `[/BAR]`.

[1] Conference and Workshop on Neural Information Processing Systems (NIPS) is one of the top conferences in Machine Learning

- **Data Analysis (5pts)** Calculate the empirical distribution over distances between `[BAR]` and `[/BAR]` in the training data. Present the histogram of this distribution and discuss it.

- **Linearly Bar-Aware: Model (10pts)** Develop a language model that assigns a probability to seeing `[/BAR]` that linearly increases with token distance to the last encounter of `[BAR]`. For all other elements of the vocabulary, it should assign uniform probability (which needs to decrease with increasing `[/BAR]` probability). This sub-task only requires you to formalize the model mathematically. What are the parameters of the model?

- **Linearly Bar-Aware: Implementation (10pts)** Code up the model you describe above, starting with the code in `BarQuestion`. Find the best parameters for the model by tuning them on perplexity results for the development set. Present in your report a graph that shows the dependency of perplexity on model parameters. Compare that with the performance of a regular uniform model.

- **Per-Distance Bar-Aware: Model (10pts)** Develop a language model that assigns a probability to seeing `[/BAR]` that again depends on the distance to the last `[BAR]`. However, this time there should be one parameter per distance. That is, $p([/\texttt{BAR}]|h) = \theta_{\text{index}_h([\texttt{BAR}])}$ where $\text{index}_h([\texttt{BAR}])$ is the index (position) of `[BAR]` in the history h. If `[BAR]` is not in the history the index should be -1. Again for all other elements of the vocabulary it should assign uniform probability (which needs to decrease with increasing `[BAR]` probability). This sub-task requires you to formalize the model mathematically and to describe how get the *Maximum Likelihood Estimate* (MLE) of the model.

- **Per-Distance Bar-Aware: Implementation (10pts)** Code up the model you describe above by adding a further LM to your code in `BarQuestion` and train it (a) using MLE and (b) by optimizing perplexity on the development set. Present and discuss the results in your report.

- **Interpolation and Start of Bars (10pts)** Improve the Per-Distance Bar-Aware LM to take into account that after seeing a `[/BAR]` a `[BAR]` follows. Then interpolate this model with a unigram model trained on your training set, and find the best interpolation weight α. Sample 1000 words from this LM and prepare a histogram of the bar length distribution on this sample. Present and discuss the results in your report.

Problem 3. Language Models for Document Classification (25pts) You want to distinguish between songs by *The Roots*, *J-Live* and *Rakim* and are given a training corpus with (labelled) songs of each artist. At test time you will be given an unlabelled song and need to predict which artist it belongs to.

- **Model (10pts)** Describe a noisy channel model of this classification task, and explain how language models can be incorporated into such a model. What language model order would you recommend, and how does this noisy channel model relate to a Naive Bayes classifier? Can you imagine any ways to improve the language model for this task (even if these would lower the models ability to generate actual language)?

- **Implementation (15pts)** Implement your language model from above in `object DocClassifyQuestion`. Present your prediction results in the report in terms of classification accuracy.

Problem 4. Ignorant Machine Translation (10pts, no coding required) This problem focuses on language models as a crucial part of Machine Translation systems. To build an MT system you also need a translation model. In the following you will develop an MT system for the most ignorant translation model ever.

You are given a rap song in German (yes, these exist!) and are supposed to translate it into English. Unfortunately your translation model $p(g|e)$ is extremely ignorant: it doesn't depend on the content of the English sentence e, but only on its length. That is, $p(g|e) = p'(g|\text{length}(e))$ for some distribution p'. Let us assume that for every German rap song g you can evaluate the function $\text{support}_{p'}(g)$ to get the set of lengths l for which $p'(g|l) > 0$.

Say you have a unigram LM, how would you find the most likely English translation of a German rap song under this LM and translation model in the noisy channel framework? How would the algorithm change for a bigram LM? How could you make it more efficient (maybe at the cost of optimality guarantees)?

Problem 5. Open-Ended Language Model Development (Optional, 10pts) Starting with the code in `object OpenEndedLMQuestion`, improve the language as you see fit in order to improve development perplexity. You are allowed to use any technique you like (larger n-grams, better smoothing, etc.), but you can only use the data we provide you. You will be scored based on your model's perplexity on a hidden test set and how you describe and motivate your approach.