Assignment 1

Introduction to NLP - Spring '26

Deadline: 11th February 2026, 23:59

General Instructions

1. The assignment must be implemented in Python, and no standard libraries for calculating n-grams, tokenization or LM should be used (for example: `nltk`, `inltk` or `spacy` for n-grams and tokenization or `pytorch` for Language Modelling are **not** to be used).
2. Your grade will depend on both the validity of your output and your interpretation of the output and method used.
3. A single .zip file needs to be uploaded to the Moodle submission portal.
4. Start early since no extension will be possible beyond the announced deadline.

1 Tokenization

Use the Mongolian and English CC-100 corpora (truncated) for this section, which can be downloaded from here (you don't actually need to know Mongolian for this assignment, but you do need to know English). For each corpus, complete the following tasks:

✓ **Task 1.1.** Clean the corpus of any detritus like excessive spacing or bad alignment or weird unicode artefacts (generally anything that can hamper the tokenizer). Also split it into three partitions: training, validation and testing. Henceforth, you must not use the validation or testing partition for training. Make note of the ratios that you partition the corpus in.

What tokenizers?

Task 1.2. For the cleaned corpus, train three kinds of tokenizers: a whitespace-based tokenizer which simply treats whitespace as a token separator (and considers special characters as tokens of their own), a regex-based tokenizer which uses a regular expression (that you define) to determine what constitutes a token, and BPE. Make note of any simplifying assumptions made.

Task 1.3. For each of the three tokenizers described above, make note of at least three examples of when a sentence's tokenization is sensible (from the perspective of language modelling) and at least three examples of when it isn't. Also explain why you think they're sensible or not. (See Resources for useful hints)

By the end of this, you should have six tokenizers (three for English and three for Mongolian) and an idea about which of these is more suited for each language. This will help for the next section.

2 Autocomplete and Language Modelling

For this section, you'll be implementing an autocomplete system (which is basically language modelling) using two different smoothing methods and analysing their behaviour. Only use the English dataset from the previous section and the tokenizers that you created for it.

Task 2.1. Implement a probabilistic 4-gram language model without using any neural networks or ML libraries. Use this language model to predict entire sentences based on partial input. This will require you to make the assumption that obtaining the most likely completed sentence is the same as iteratively obtaining the most likely token until a sentence ends. Feel free to make any further simplifying assumptions here, but do make note of them.

Task 2.2. Implement two kinds of smoothing for this model: Witten-Bell smoothing and Kneser-Ney smoothing. Finally, you should have 9 different LM results: one for each kind of tokenizer and each smoothing method (including no smoothing).

Task 2.3. Report the perplexity of each of these models on the test partition.

Task 2.4. Examine the behaviour of these 9 autocomplete models and qualitatively analyse and compare each one's functioning. Specifically, make note of situations where each one behaves correctly/incorrectly, and explain why they're happening (at least three each).

3 Submission format

Make sure to submit a compressed version of the following file structure.

```
<roll_number>_A1.zip
|
|-- tokenizers.py
|-- language_models.py
|-- utils.py (optional, for any big helper functions)
|-- README.md (mandatory)
|-- report.pdf (mandatory)
|-- ...
```

In the above tasks, any explanations, examples or assumptions that were required must be included in the report. Any deviations from this file structure (discouraged) must also be explained in the `README` file (a markdown file isn't necessary, any relevant file format will do). Instructions to run your code must be included in the `README` and must be correct. Write clean code and **do not submit Jupyter Notebooks**. If you use one, make sure to convert it to a Python script before submitting. No Jupyter Notebooks will be evaluated.

Resources

1. CS224N – Language Modelling and Smoothing
2. Bill McCartney – NLP Lunch Tutorial: Smoothing
3. J&M Chapter 3 – N-gram Language Models
4. J&M Appendix B – Kneser-Ney Smoothing
5. Wikipedia – Mongolian Language (you might need some basic information about the writing and grammar for Task 1.3).
6. Slides for Basic Regex