Assignment 1: Language Modelling and Smoothing

Course: Introduction to Natural Language Processing

Deadline: January 23rd, 2025 | 23:59

General Instructions

1. The assignment must be implemented in Python.
2. No standard library for calculating n-grams or LMs should be used.
3. You are allowed to use AI tools or any internet resources as long as your implementation is in accordance with the assignment guidelines.
4. A single .zip file needs to be uploaded to the Moodle Course Portal.
5. Please start early since no extension to the announced deadline would be possible.

1 Tokenization

You have been given two corpora for cleaning. Your task is to design a tokenizer using regex or any standard libraries. This tokeniser will be used later for smoothing and language modelling.

You are also encouraged to try other tokenization and placeholder substitution schemes based on your observations from the corpora used for the smoothing task to achieve a better language model. You may find URLs, hashtags, mentions, percentages, age values, expressions indicating time, time periods occurring in the data.

You're free to explore and add multiple such reasonable tokenization schemes in addition from the ones listed above. Specify any such schemes you use in the final README.

2 N-Grams

Design a function in Python that takes a value of 'N' and the <corpus_path> and generates an N-sized N-gram model from both the given corpus.

3 Smoothing and Interpolation

You have been given two corpora: "Pride and Prejudice" and "Ulysses". Your task is to design Language Models for both using smoothing techniques. Be sure to use the tokenizer created in the Tokenization task.

1. Create language models with the following parameters on each of the corpus:
    (a) On "Pride and Prejudice" corpus for N=1,3 and 5:
        i. LM 1: Tokenization + N-gram LM + Laplace Smoothing
        ii. LM 2: Tokenization + N-gram LM + Good-Turing Smoothing
        iii. LM 3: Tokenization + N-gram LM + Linear Interpolation

    (b) On "Ulysses" corpus for N=1,3 and 5:
        i. LM 4: Tokenization + N-gram LM + Laplace Smoothing
        ii. LM 5: Tokenization + N-gram LM + Good-Turing Smoothing
        iii. LM 6: Tokenization + N-gram LM + Linear Interpolation

2. For each of these corpora, create a test set by randomly selecting 1000 sentences. This set will not be used for training the LM.
    (a) Calculate perplexity score for each sentence of "Pride and Prejudice" corpus and "Ulysses" corpus for each of the above models and also get average perplexity score on the train corpus.

    (b) Report the perplexity scores for all the sentences in the training set. Report the perplexity score on the test sentences as well, in the same manner above

Note:

1. Laplace Smoothing(add one) and Good-Turing Smoothing are techniques used to adjust the probability distribution of events in N-gram models, while Linear Interpolation is a method of combining different N-gram models (like unigram, bigram, trigram) to get a better estimate of the probabilities.

2. Use the formula given below to implement Good Turing smoothing.

$$P_{GT}(w_1 \dots w_n) = \frac{r^*}{N}$$

   where:

$$r^* = \frac{(r+1)S(r+1)}{S(r)}$$

   Here, the notation $S(\cdot)$ means the smoothed function. For small values of r, it is reasonable to set $S(N_r) = N_r$, that is, no smoothing is performed. For large values of r, values of $S(N_r)$ are read off the regression line given by the logarithmic relationship is:

$$\log(N_r) = a + b \log(r)$$

   Here, Nr represents the number of times n-grams of frequency r have occurred.

   For unseen events:

$$P_{GT}(w_1 \dots w_n) = \frac{N_1}{N}$$

For more details, refer to the resources section at the end.

4 Generation

You should be able to generate text from a given input using each of the Language Models that you've developed. This task is commonly known as Next Word Prediction.

This means, that for a given language model, when you give it an input, it'll use the input context to generate the most likely word that comes next, or the word with the highest probability.

1. Using the generated N-gram models (without the smoothing techniques), try generating sequences. Experiment with different values of N and report which models perform better in terms of fluency.

2. Attempt to generate a sentence using an Out-of-Data (OOD) scenario with your N-gram models. Analyze and discuss the behavior of N-gram models in OOD contexts.

3. Now try to generate text using the models with the smoothing techniques (LM1, LM2, LM3, LM4, LM5, LM6) for N=1,3 and 5 each.

Provide the analysis along with the examples for each of the points in the Report.

5 Corpus

The following corpora have been given to you for training:

1. Pride and Prejudice corpus (1,24,970 words)

2. Ulysses Corpus (2,68,117 words)

Please download the corpus files from this link.

6 Submission Format

Zip the following into one file and submit in the Moodle course portal. Filename should be <roll_number>_<assignment1>.zip, eg: 2022xxxxxx_assignment1.zip:

1. Source Code for Tokenization:

```
tokenizer.py
your text: Is that what you mean? I am unsure.
tokenized text: [['Is', 'this', 'what', 'you', 'mean', '?'],
['I', 'am', 'unsure', '.']]
```

   On running this file, the expected output is a prompt which asks for an input and outputs the tokenized sentences.

2. Source Code for Language Models:

   ```
   language_model.py <lm_type> <corpus_path>
   ```

   LM type can be l for Laplace Smoothing, g for Good-Turing Smoothing Model and i for Interpolation Model.
   On running the file, the expected output is a prompt, which asks for a sentence and provides the probability of that sentence using the given mechanism. Therefore, an example would be:

   ```
   python3 language\_model.py i ./corpus.txt
   input sentence: I am a woman.
   score: 0.69092021
   ```

3. Source Code for Generation:

   ```
   generator.py <lm_type> <corpus_path> <k>
   ```

   LM type can be l for Laplace Smoothing, g for Good-Turing Smoothing Model and i for Interpolation Model.
   k denotes the number of candidates for the next word to be printed.
   On running the file, the expected output is a prompt, which asks for a sentence and outputs the most probable next word of the sentence along with it's probability score using the given mechanism. Therefore, an example would be:

   ```
   python3 generator.py i ./corpus.txt 3
   input sentence: An apple a day keeps the doctor
   output:
   away 0.4
   happy 0.2
   fresh 0.1
   ```

4. Report containing the perplexity scores of all LMs and your analysis of the results in a PDF:

   (a) For each LM, submit a text file with perplexity scores of each sentence in the following format:

      avg_perplexity

      sentence_1 <tab> perplexity

sentence_2 <tab> perplexity

(b) Naming must be:

<roll number>_LM1_N_train-perplexity.txt,

<roll number>_LM1_N_test-perplexity.txt

`Note:` Don't submit perplexity scores on dev/val splits.

5. README on how to execute the files, how to get perplexity of sentences along with any other information

6. In total, you will have:

    (a) 36 text files - one for each of the Language Model and Corpus Combination
    (b) 3 .py files
    (c) Report
    (d) README

7 Grading

Evaluation will be individual and will be based on your viva, report and submitted code review. You are expected to walk us through your code, explain your experiments, and report. You will primarily be graded based on your conceptual understanding.

1. Tokenization (5)

2. N-grams (5)

3. Smoothing and Interpolation

    (a) Laplace (5)
    (b) Good Turing (10)
    (c) Interpolation (10)

4. Generation and Analysis (20)

5. Viva during Evaluation (45)

8 Resources

1. An Introduction to N-Gram Language Modelling and Methods
2. Paper to help with lambda calculation for Interpolation
3. Slides for Language Modelling and Good Turing
4. Paper on Good Turing Smoothing