CS7.401: Introduction to NLP | Assignment 1

Instructor: Manish Shrivastava

Deadline: February 11, 2023 | 23:59

General Instructions

1. The assignment must be implemented in Python.
2. No standard library for calculating n-grams, tokenization or LMs should be used.
3. Submitted assignment must be your original work. Please do not copy any part from any source including your friends, seniors, and/or the internet. If any such attempt is caught, then serious actions including an F grade in the course is possible.
4. A single .zip file needs to be uploaded to the Moodle Course Portal.
5. Your grade will depend on the correctness of answers and output. In addition, due consideration will be given to the clarity and details of your answers and the legibility and structure of your code.
6. Please start early since no extension to the announced deadline would be possible.

1 Tokenization

You have been given two corpuses for cleaning. Your task is to design a tokenizer using regex, which you will later use for smoothing and language modelling as well.

1. Create a Tokenizer to handle following cases:
   (a) Word Tokenizer
   (b) Punctuation
   (c) URLs
   (d) Hashtags (#manchesterisred)
   (e) Mentions (@john)
2. For the following cases, replace the tokens with appropriate placeholders:
   (a) URLs: <URL>
   (b) Hashtags: <HASHTAG>
   (c) Mentions: <MENTION>

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 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 Smoothing

You have been given two corpus: “Pride and Prejudice” corpus, and “Ulysses” corpus. Your task is to design Language Models for both corpora using smoothing. Ensure that you use the tokenizer created in task 1 for this task.

1. Create language models with the following parameters:
   (a) On “Pride and Prejudice” corpus:
      i. LM 1: tokenization + 4-gram LM + Kneser-Ney smoothing.
      ii. LM 2: tokenization + 4-gram LM + Witten-Bell smoothing.
   (b) On “Ulysses” corpus:
      i. LM 3: tokenization + 4-gram LM + Kneser-Ney smoothing.
      ii. LM 4: tokenization + 4-gram LM + Witten-Bell smoothing.
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.

3 Neural Language Model

You have been given two datasets. Make your own data splits (for eg: 70%, 15%, 15%). Train separates models on each corpus.

1. Create a neural language model using a recurrent architecture such as LSTMs for both the corpora provided.
   (a) Use the train split for training the model parameters.
   (b) Dev split for hyperparameter optimization as well saving the best model checkpoint.
   (c) Test split for a final check on the performance.
2. Perplexity computation:
   (a) Calculate the perplexity score for each sentence of the train splits for both the corpora only using the trained model and get the averaged perplexity score on it.
   (b) Report the perplexity score on the test sentences as well, in the same manner as above {LM 5 (Pride and Prejudice), LM 6 (Ulysses)}.
   (c) Compare and analyse the behaviour of the different LMs and put your analysis and visualisation in report.

4 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.

5 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:

(a) language_model.py: Runs the language model given the following:

$ python3 language_model.py <smoothing type> <path to corpus>

Smoothing type can be k for Kneser-Ney or w for Witten-Bell. On running the file, the expected output is a prompt, which asks for a sentence and provides the probability of that sentence using the two smoothing mechanisms. Therefore, an example would be:

$ python3 language_model.py k ./corpus.txt

input sentence: I am a woman.

0.69092021

(b) neural_language_model.py: Runs the language model given the following:

$ python3 neural_language_model.py

On running the file, the expected output is a prompt, which asks for a sentence and provides the probability of that sentence. Therefore, an example would be:

$ python3 neural_language_model.py ../models/trained_nlm.pth

input sentence: I am a woman.

0.69092021

2. 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_train-perplexity.txt,

<roll number>_LM1_test-perplexity.txt

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

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

4. In total, You will have:

   (a) 12 text files:

   For smoothing techniques:

   i. <roll_number>_LM1_train-perplexity.txt
   ii. <roll_number>_LM1_test-perplexity.txt
   iii. <roll_number>_LM2_train-perplexity.txt
   iv. <roll_number>_LM2_test-perplexity.txt
   v. <roll_number>_LM3_train-perplexity.txt
   vi. <roll_number>_LM3_test-perplexity.txt
   vii. <roll_number>_LM4_train-perplexity.txt
   viii. <roll_number>_LM4_test-perplexity.txt

   For neural language model:

   ix. <roll_number>_LM5_train-perplexity.txt
   x. <roll_number>_LM5_test-perplexity.txt
   xi. <roll_number>_LM6_train-perplexity.txt
   xii. <roll_number>_LM6_test-perplexity.txt

   (b) 2 .py files (Smoothing and NLM).
   (c) 2 model .pt/.pth files (one for each corpora)
   (d) Report
   (e) Readme

6 Grading

Evaluation will be individual and will be based on your viva, report, submitted code review. In the slot you are expected to walk us through your code, explain your experiments, and report. You will be graded based on correctness of your code, accuracy of your results and quality of the code.

Marks Distribution (out of 100):

1. Tokenization: 5 Marks.
2. Smoothing: 40 Marks
   (a) Kneser–Ney: 20 Marks
   (b) Witten-Bell: 20 Marks
3. Neural Language Model: 30
4. Quality and Efficiency: 5 Marks
5. Viva during Evaluation: 20 Marks

7 Resources

1. Chapter 3 of the Speech and Language Processing book by Jurafsky & Martin
2. Chapter 9 of the Speech and Language Processing book by Jurafsky & Martin
3. Understanding LSTM Networks
4. An Empirical Study of Smoothing Techniques for Language Modeling for Witten-Bell Smoothing
5. Official Python documentation and testing playground for regex.

8 FAQs

1. **Can I submit my code in Jupyter Notebook?**

   No, the final submission should be a Python script. You may work using Jupyter Notebooks, but make sure to convert them to .py files before submitting.

2. **Do I need to code everything from scratch?**

   No. You are free to use neural and LSTM layers from deep learning frameworks like PyTorch / Keras etc.

3. **My model takes too long for train/test. What should I do?**

   Ensure you are using a GPU environment with proper batch sizes for training. For computing answer given an input, ensure you are storing the weights and not calculating them while running the script.

4. **My model size is greater than the limit allowed on moodle. What should I do??**

   Place the read-only link in the readme file after uploading the model to any cloud storage. Mention every specific step involved in running the model.