INTERNATIONAL INSTITUTE OF
INFORMATION TECHNOLOGY
HYDERABAD

Intoduction To NLP 2023 : Assignment 1

Course Name : Introduction to NLP
Course Code : CS7.401
Semester : Spring'23
Instructor Name : Prof. Manish Shrivastava

Sk. Abukhoyer
Roll : 2021201023
Mtech CSE

IIIT Hyderabad

February 18, 2023

Contents

1 Abstract

This report is about how assignment has been done. In this assignment we are asked to do mainly three things, One is tokenization which is a preprocessing part of the text files, second part is building two 4-Gram language model using Kneser-Ney smoothing and Witten-Bell smoothing, last part of this assignment is creating a neural language model using LSTM. A language model is a probability distribution over strings that describes how often the string s occurs as a sentence in some domain of interest. Language models are employed in many tasks including speech recognition, optical character recognition, handwriting recognition, machine translation, and spelling correction.

2 Tokenization

Tokenization is the first step in any NLP pipeline. It has an important effect on the rest of your pipeline. A tokenizer breaks unstructured data and natural language text into chunks of information that can be considered as discrete elements. The token occurrences in a document can be used directly as a vector representing that document.

This immediately turns an unstructured string (text document) into a numerical data structure suitable for machine learning. They can also be used directly by a computer to trigger useful actions and responses. Or they might be used in a machine learning pipeline as features that trigger more complex decisions or behavior.

2.1 Procedure

I have been given two corpuses which are "Pride and Prejudice" and "Ulysses" for cleaning. My task is to design a tokenizer using regex, which I will use later in next two part of this assignment in smoothing and language modelling as well. The algorithm I have implemented for tokenization which will be applied for each sentence in the corpus is following:

- Using regex finding out the pattern for punctuation, URLs, Hashtags and Mentions
- In the next step I removed all the punctuation symbols except hashtags and mention
- URLs are replaced by $\langle URL \rangle$
- Hashtags are replaced by $\langle HASHTAG \rangle$
- Mentions are replaced by $\langle MENTION \rangle$
- In the final step return the splitted sentence as a list of words

Tokenization can separate sentences, words, characters, or subwords. When we split the text into sentences, we call it sentence tokenization. For words, we call it word tokenization.

3 Smoothing

the number of times the word occurs in some corpus of training data divided by the number of times the word occurs. This value is called the maximum likelihood (ML) estimate. Unfortunately, the maximum likelihood estimate can lead to poor performance in many applications of language models. Smoothing is used to address this problem. The term smoothing describes techniques for adjusting the maximum likelihood estimate of probabilities to produce more accurate probabilities. The name comes from the fact that these techniques tend to make distributions more uniform, by adjusting low probabilities such as zero probabilities upward, and high probabilities downward. Not only do smoothing methods generally prevent zero probabilities, but they also attempt to improve the accuracy of the model as a whole.

3.1 Kneser-Ney Smoothing

Kneser and Ney have introduced an extension of absolute discounting where the lower-order distribution that one combines with a higher-order distribution is built in a novel manner. we can mathematically motivate their algorithm by selecting the lower-order distribution such that the marginals of the higher-order smoothed distribution match the marginals of the training data. As in absolute discounting, let $0 \leq D \leq 1$. Then, we assume that the model has the form given in Equation

$$\underline{P_{KN}(w_i \mid w_{i-n+1}^{i-1})} = \frac{\max(\underline{c_{KN}}(w_{i-n+1}^{i}) - d, 0)}{\underline{\underline{c_{KN}}(w_{i-n+1}^{i-1})}} + \underline{\lambda(w_{i-n+1}^{i-1}})P_{KN}\underline{(w_i \mid w_{i-n+2}^{i-1})})$$

3.2 Witten-Bell Smoothing

Witten–Bell smoothing was developed for the task of text compression, and can be considered to be an instance of Jelinek–Mercer smoothing. In particular, the nth-order smoothed model is defined recursively as a linear interpolation between the nth-order maximum likelihood model and the (n-1)th-order smoothed model. To compute the parameters $\lambda_{w_{i-n+1}^{i-1}}$ for Witten–Bell smoothing, we will need to use the number of unique words that follow the history w_{i-n+1}^{i-1} .We will write this value as $N_{1+}(w_{i-n+1}^{i-1})$, formally defined as

$$N_{1+}(w_{i-n+1}^{i-1}\bullet) = |\{w_i : c(w_{i-n+1}^{i-1} w_i) > 0\}|.$$

The mathematical equation for Witten–Bell smoothing is defined as

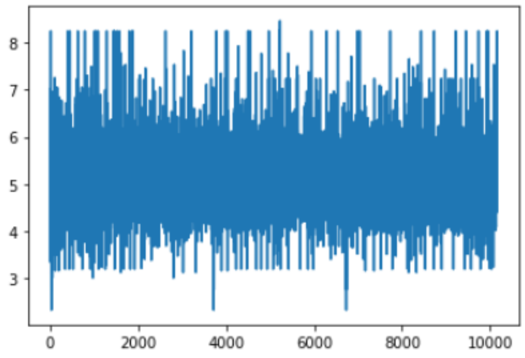
$$p_{\mathrm{WB}}(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i}) + N_{1+}(w_{i-n+1}^{i-1}\bullet)\, p_{\mathrm{WB}}(w_i|w_{i-n+2}^{i-1})}{\sum_{w_i} c(w_{i-n+1}^{i}) + N_{1+}(w_{i-n+1}^{i-1}\bullet)}.$$

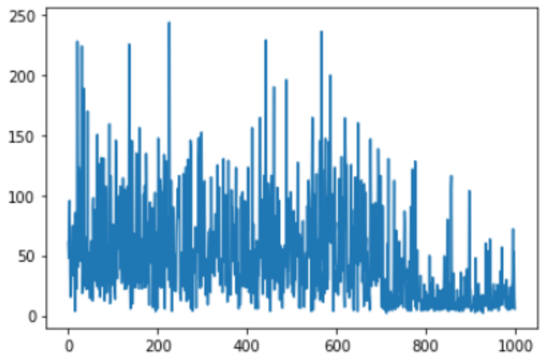
3.3 Analysis

The factor with the largest influence is the use of a modified lower-order distribution as in Kneser–Ney smoothing. This seemed to be the primary reason that the variations of Kneser–Ney smoothing performed so well relative to the remaining algorithms.

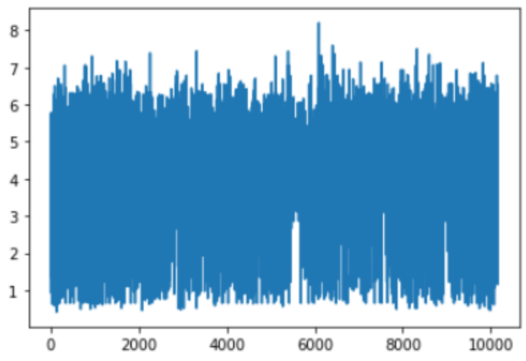
Visualization for the below result files:

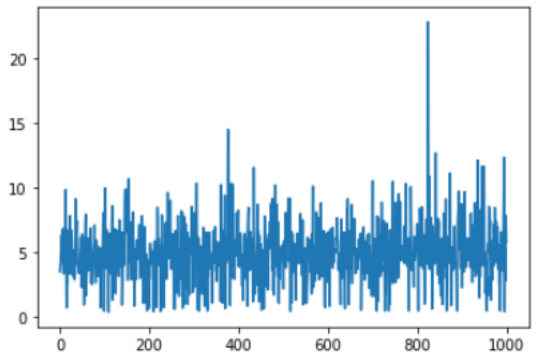
2021201023_LM1_train-perplexity.txt and 2021201023_LM1_test-perplexity.txt



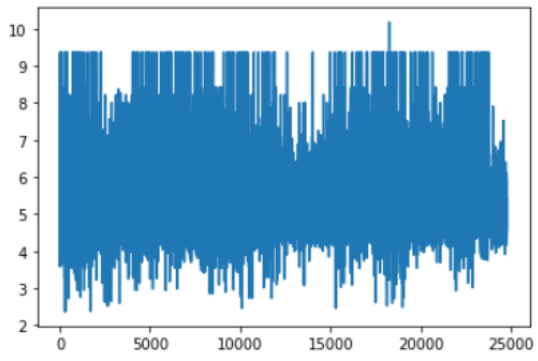


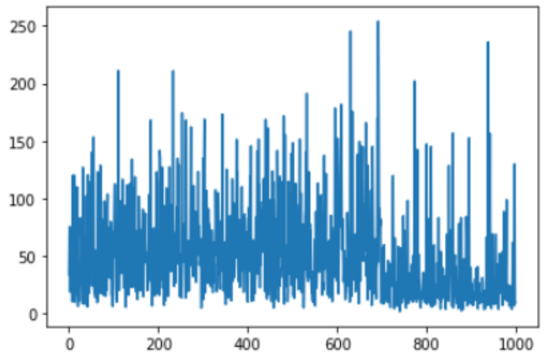
2021201023_LM2_train-perplexity.txt and 2021201023_LM2_test-perplexity.txt



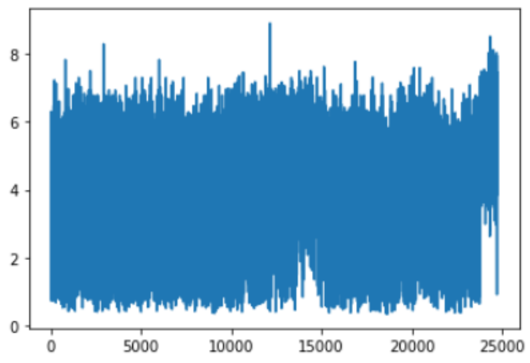


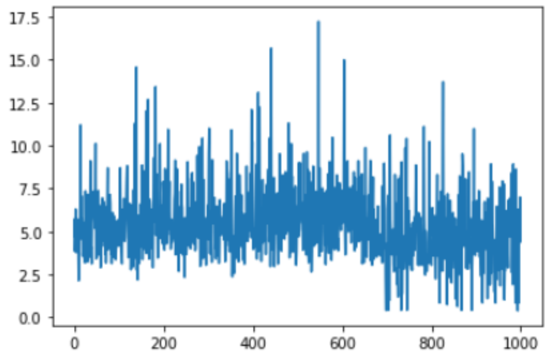
2021201023_LM3_train-perplexity.txt and 2021201023_LM3_test-perplexity.txt





2021201023_LM4_train-perplexity.txt and 2021201023_LM4_test-perplexity.txt





4 Neural Language Model

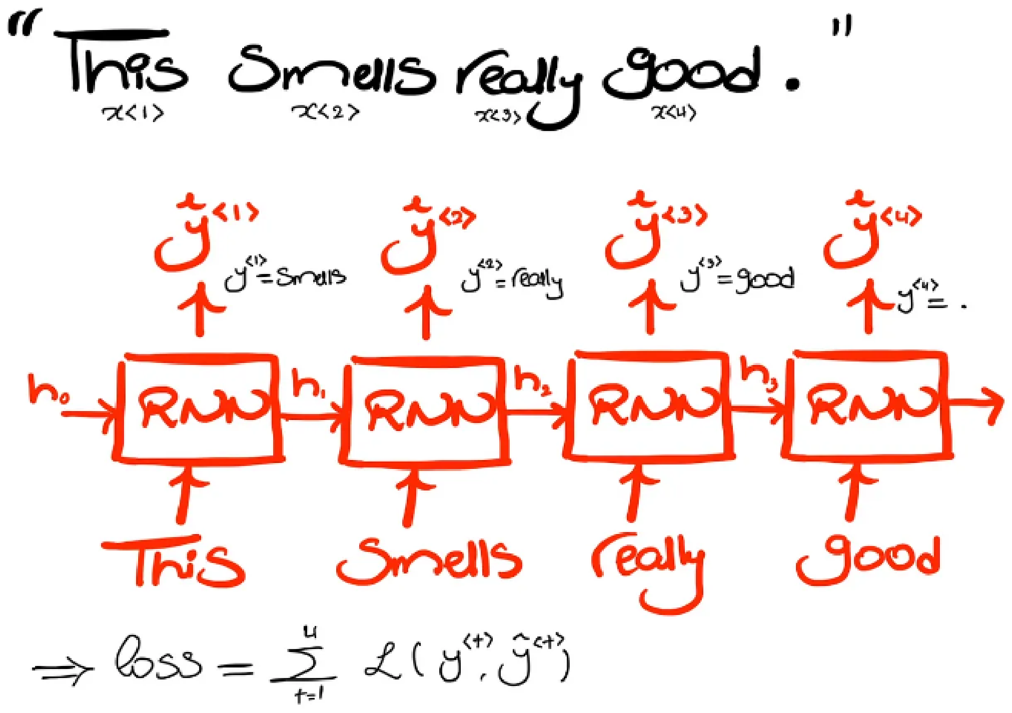
Language models are statistical models that are trained on large amounts of text data, and they are designed to predict the likelihood of a sequence of tokens. This means that given a particular sequence of words, a language model can estimate the probability of the next word in the sequence. Language models can be used for a variety of tasks, such as text generation, machine translation, and speech recognition. In text generation, the model can be used to generate new text by repeatedly predicting the next word based on the previously generated text. This process is called sequence generation, and it allows the model to generate sequences of text that are similar to the training data, but not identical.

4.1 Approach

Firstly, we need to split the datasets into train, valid, and test sets. Let's assume that we have two datasets: "Pride and Prejudice" and "Ulysses". We will split each dataset into train, dev, and test sets with a split ratio of 70

We can perform the following steps to develop a neural language model utilising LSTMs as a recurrent architecture:

1. **Step 1: Prepare data**
   Organize the two datasets into three groups: Train, Valid, and Test. Then, load the three groups. Map every token to a different integer to tokenize each dataset and produce a vocabulary. A well-liked NLP library, such NLTK or Spacy, has


" This Smells really good . "
$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ $x^{<4>}$
$\hat{y}^{<1>}$ $y^{<1>}$ = Smells
$\hat{y}^{<2>}$ $y^{<2>}$ = really
$\hat{y}^{<3>}$ $y^{<3>}$ = good
$\hat{y}^{<4>}$ $y^{<4>}$ = .
h_0 h_1 h_2 h_3
RNN RNN RNN RNN
This Smells really good
$\Rightarrow loss = \sum_{t=1}^{4} \mathcal{L}(y^{<t>}, \hat{y}^{<t>})$

a tokenizer that we can utilise. The vocabulary should be used to transform the tokenized sequences into an integer sequence. The zero padding will equalise the length of the sequences. For every dataset split, create data loaders.

```
# tokens contains the list of sentence list
tokens = load_dataset("pride_and_prejudice.txt")
X_train, X_rem= train_test_split(tokens, train_size=0.7)
X_valid, X_test= train_test_split(X_rem, test_size=0.5)
```

2. **Step 2: Creating the Model**
   Make an LSTM model with an output fully connected layer, an embedding layer, a predetermined number of LSTM layers, and a hidden size. The loss function should be cross-entropy loss. Employ Adam optimizer or stochastic gradient descent for training.
3. **Step 3: Hyperparameter optimization and training**
   The model is trained on the train split, then evaluated on the development split. For hyperparameter customization, such as adjusting the number of layers, hidden size, and learning rate, use the dev split. On the development set, save the best model checkpoint.
4. **Step 4: Testing and proplexity computation**
   On the test split, run the saved model checkpoint test and determine the perplexity score for each sentence. Calculate the combined train and test splits' average perplexity score. Analyze how each LM behaves and write a report with your findings.

4.2 Analyse