Name: Prerana Chakraborty

Email address: prerana.chaks@gmail.com

Contact number: +916290870350

Anydesk address: **338 565 775**

Years of Work Experience: **0**

Date: **14th March 2021**

Self Case Study -2: *** Next Word Prediction using Swiftkey Data ***

"After you have completed the document, please submit it in the classroom in the pdf format."

Please check this video before you get started:
https://www.youtube.com/watch?time_continue=1&v=LBGU1_JO3kg

Overview

*** Write an overview of the case study that you are working on. ***(MINIMUM 200 words)*** ***

Introduction

This is the 2nd Self Case Study of Applied AI Course . The problem statement is **Next Word Prediction using Swiftkey Data** . Next Word Prediction is the task of predicting the next word based on a sequence of words.It is also known as Language Modeling.

Business Problem

This problem falls in the area known as natural language processing(NLP) and text mining.NLP is a branch of AI that helps computers understand , interpret and manipulate human languages. Text Mining is an AI technology that uses NLP to

convert unstructured text data from documents to structured data that can be used by ML algorithms. Next word prediction problem uses the above technologies to accomplish results.

Why next word prediction is important?

It helps in minimizing keystrokes which in turn saves time while typing, checks spelling errors and helps in word recall. Students who are still working on their language skills can take its help. Also people with dyslexia can be benefitted by this.

ML Formulation

Data Overview

Data source: https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip

This zip file contains 4 folders each containing data from 4 different languages namely English, Russian , Finnish and German. I will be using the data from English folder . It contains the following files.

- en_US.twitter.txt - This text file contains all English language data from twitter.
- en_US.news.txt - This text file contains all English language data from news websites and channels.
- en_US.blogs.txt - This text file contains all English language data from all blog sites.

This data is provided by Swiftkey.

Mapping Real-World to ML Problem

Next word prediction involves predicting the next word . So given a sequence of words generated from the corpus , I have to predict the next word which has highest probability of occurrence.

Thus it is a predictive modeling problem for languages also known as Language Modeling.

We can also approach this problem in another way. We can consider each of the next word to be predicted as a class. So it can be treated as MultiClass Classification problem.

Business Objectives and Constraints

Objectives

- Cleaning text data from the corpus
- Creating Sequences from the cleaned data
- Building Statistical or Neural Language Models to predict next word

Constraints

- Memory Constraint - Corpus size may be too large which might cause a memory error .
- Latency Constraint - It is a low latency problem as the entire problem is designed to enable fast typing.
- OOV words- It is important to take care of out of vocabulary words because all words may not be present in the corpus but the model should be able to handle it.

Performance Metric

The metric for this problem is Categorical Cross Entropy.

Each next word to be predicted is considered a category so I will be using Categorical Crossentropy for this.

Formula for Categorical CrossEntropy : $-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$

M- Number of categories
p - predicted probability observation o is of class c

Research-Papers/Solutions/Architectures/Kernels

*** Mention the urls of existing research-papers/solutions/kernels on your problem statement and in your own words write a detailed summary for each one of them. If needed you can include images or explain with your own diagrams. ***

1. Paper doi:10.1109/icdse47409.2019.8971796

   This research paper is on Next Word Prediction in Hindi using Deep Learning Techniques

   I have summarised the techniques that has been used for next word prediction for Hindi Language.

   - Text is cleaned and unique words are generated
   - These unique words are stored in a dictionary and mapped to indices as neural networks work better with indices
   - Sequence Lengths are set , for eg. if sequence length=6 then sentence is divided into 6:1 ratio ie. first six words in input_x and the remaining in input_y
   - Tensors are created from input_x based on the sequence length and stored in data_x
   - For each of the tensor sequences in data _x , a sequence of next words that is input_y is stored as a tensor in data_y.
   - LSTM and Bi-LSTM are used for the prediction based on the highest probability of the next word.

- Activation function used is Softmax to assign probability of the next words .
- As this problem is treated as a multiclass classification problem so the loss function is categorical cross entropy.
- The LSTM model ran for 197 epochs and the Bi-LSTM model ran for 121 epochs .
- It has been repeated for all sequence lengths.

2. Paper doi: https://doi.org/10.1007/s00521-020-05245-3(0123456789().,-volV)(0123456789(). ,- volV)

   This paper develops next word prediction model for Kurdish Language. Here they have discussed the N-Gram language model for this problem.

   N-Gram Language Model

   In language models , either probabilities are assigned to a series of words or probabilities are assigned to the next word given some preceding words.

   Here **2 scenarios** are discussed:

   1st Scenario- In case of a sentence (w_1 w_2 w_3 w_4 w_5) ,the probability for the sentence is given by $P(w_1)*P(w_2)*P(w_3)*P(w_4)*P(w_5)$ where $P(w_n)$ is the probability of w_n .

   $P(w_n)$= (Number of occurrences of wn in the corpus) / (Total number of words in the corpus)

   2nd Scenario - In case of a sentence (w_1 w_2 w_3 w_4 w_5) ,the probability for the sentence is given by $P(w_1)*P(w_2|w_1)*P(w_3|w_1w_2)*P(w_4|w_1w_2w_3)*P(w_5|w_1w_2w_3w_4)$ where $P(w_n|w_{n-1})$ is the probability of wn given the preceding word is w_{n-1} .

   $P(w_n|w_n\text{-}1)= P(w_{n-1}\ w_n)/P(w_{n-1})$ that is Probability of occurrence of (w_{n-1} w_n) in the corpus divided by Probability of occurrence of w_{n-1}.

N- Gram models can be of different types . Those are unigram , bigram , trigram and so on.

Unigram Model - In this model, the probability of each word in the sentence is given by $P(w_i)$ = Count of w_i in the text corpus / total words .

Bigram Model - This is similar to the 2nd scenario discussed above . But here we take into account only the previous word.

In the case of a sentence (w_1 w_2 w_3 w_4 w_5) ,the probability for the sentence is given by $P(w_1)*P(w_2|w_1)*P(w_3|w_2)*P(w_4|w_3)*P(w_5|w_4)$ where $P(w_n|w_{n-1})$ is the probability of w_n given the preceding word is w_{n-1} .

Trigram Model - This is similar to the bigram model . But here we take into account 2 previous words.

Thus the expression for n-gram model is

$P(w_i|w_1w_2...w_{i-1})=P(w_i \mid w_{(i-n+1)}....w_{i-1})$.

This is also called Markov assumption.

Now to deal with 0 probability , **Stupid Backoff Algorithm** is used.

If higher order n-gram results in 0 , then we backoff to lower order n-gram.

It is given by the following formula.

$$S(w_i|w_{i-k+1}^{i-1}) = \begin{cases} f(w_{i-k+1}^{i}) / f(w_{i-k+1}^{i-1}) & \text{if } f(w_{i-k+1}^{i}) > 0 \\ \alpha S(w_i | w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

For example: Let us consider a sequence of words “this is a very beautiful” .

Here we have to find probability of “beautiful” given “this is a very”. Let's say "beautiful" never occurred in the context "this is a very" so for the

4-grams model "beautiful" has probability 0 . So we backoff to 3-gram model and find the probability as α*P(“beautiful”|”is a very”).

3. **Article on Smoothing**

   Link:
   https://towardsdatascience.com/n-gram-language-models-af6085435eeb

   Smoothing is required to deal with words that appear in the test set in an unknown context .There are different Smoothing techniques .

- Laplace Smoothing : For calculating probability , we simple add 1 to the numerator and an additional V vocabulary to the denominator.
  P(word)=(wordcount+1)/(totalnumberofwords(N)+V)

  P(new_word)=1/(N+V)

- Add k- Smoothing : Instead of adding 1 to the frequency of the words , we will be adding some fraction (k) to the count.

  P(word)=(wordcount+k)/(totalnumberofwords(N)+kV)

  P(new_word)=k/(N+kV)

- Backoff and Interpolation : Backoff is same as the Stupid Backoff algortthm explained earlier. Interpolation is the way of combining different n-gram models such as unigram,bigram , trigram models by giving weights to each of them.

$$\begin{aligned}\hat{P}(w_n|w_{n-2}w_{n-1}) &= \lambda_1 P(w_n|w_{n-2}w_{n-1}) \\ &\quad +\lambda_2 P(w_n|w_{n-1}) \\ &\quad +\lambda_3 P(w_n)\end{aligned}$$

First Cut Approach

*** Explain in steps about how you want to approach this problem and the initial experiments that you want to do. ***(MINIMUM 200 words)*** ***

- Reading the data from 3 different text files and then appending them together into 1.
- Cleaning the text which involves removing punctuations , numbers,special characters, extra whitespaces .
- Stopwords won't be removed as they play an important role for next word prediction.
- Converting all words to lower case.
- **Visualizations**
    - Generating unigrams , bigrams , trigrams along with their frequency.
    - Visualizing wordclouds of n grams.
- **Featurization**
    - For Markov Chain Model using ngrams
        - I will create a dictionary with 2 keys -first key will be a tuple of different word sequences and 2nd key will have the next word.
        - The value of this dictionary will be the probability of occurrence of 2nd key given 1st key along with laplace smoothing.
        - This will be for bigram and trigram.
    - For neural network model
        - Encode the entire data to indices .
        - I will generate sequences of length 2 , 3 and 4 from the encoded data maintaining the order of indices.
        - I will then split the sequences as x and y such that only the right most index is in y rest in x .
- **Modelling**

- Markov Chain Model
  - I will develop a Markov model with Bigram and Trigram probabilities using the entire data.
- Neural Network Model
  - Divide x and y into train and test dataset for each sequence length
  - Train different LSTM models for different sequence lengths.
  - Different LSTM models will be used for different input lengths.

Notes when you build your final notebook:

1. You should not train any model either it can be a ML model or DL model or Countvectorizer or even simple StandardScalar
2. You should not read train data files
3. The function1 takes only one argument “X” (a single data points i.e 1*d feature) and the inside the function you will preprocess data point similar to the process you did while you featurize your train data
    a. Ex: consider you are doing taxi demand prediction case study (problem definition: given a time and location predict the number of pickups that can happen)
    b. so in your final notebook, you need to pass only those two values
    c. def final(X):

preprocess data i.e data cleaning, filling missing values etc

compute features based on this X

use pre trained model

return predicted outputs

final([time, location])

   d. in the instructions, we have mentioned two functions one with original values and one without it

   e. final([time, location]) # in this function you need to return the predictions, no need to compute the metric

   f. final(set of [time, location] values, corresponding Y values) # when you pass the Y values, we can compute the error metric(Y, y_predict)

4. After you have preprocessed the data point you will featurize it, with the help of trained vectorizers or methods you have followed for your train data

5. Assume this function is like you are productionizing the best model you have built, you need to measure the time for predicting and report the time. Make sure you keep the time as low as possible

6. Check this live session: https://www.appliedaicourse.com/lecture/11/applied-machine-learning-online-course/4148/hands-on-live-session-deploy-an-ml-model-using-apis-on-aws/5/module-5-feature-engineering-productionization-and-deployment-of-ml-models