Natural Language Processing

N-GRAM Learning Model

Introduction

Traditionally, we can use n-grams to generate language models to predict which word comes next given a history of words. In this assignment, We'll use the nltk library from sklearn to understand how language modeling is done.

Importing Libraries

The major libraries used in this assignment are listed and explained below :

1. re - This library is used for extracting data from a text using regular expressions
2. nltk - This is a Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP). It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning.

Dataset and Data Preprocessing

The dataset used in this assignment is a part of text from the book “Alice in the Wonderland”. This text contains 972 lines or sentences with around 20,000 words present in the corpus.

After uploading this dataset, this dataset is preprocessed by using the ‘re’ library and other inbuilt python functions.

1. The ‘re’ library is used to remove unwanted characters, spaces and symbols.
2. Then the next step is to convert the whole text to lowercase using the text.lower() function, to prevent the confusion of words beginning with Upper case and those beginning with lower case.
3. Tokenizing the Remaining Words: The remaining words are tokenized and stored in an array.
4. Lemmatization: After tokenizing the remaining words all the words are run through the Lemmatizer to group together the different inflected forms of a word so they can be analyzed as a single item.

Creating a Language Model using Dictionary, Trigrams, and calculating the word probabilities.

Now we train a trigram Language model using the “nltk.ngrams()” function where the value of the parameter “n” is “3” since we are training a trigram model. After training the trigram model we calculate the Conditional Frequency Distribution of the trigram words or subsets using the “ConditionalFreqDist()” function. Conditional frequency distributions are used to record the number of times each sample occurred, given the condition under which the experiment was run.

cfdist[(w1, w2)][w3] - calculates the probability of w3 given w1 and w2

The calculated conditional Frequency is now converted to probabilities by dividing the trigrams or cfdist[(w1, w2)][w3] by the total occurrences of w1 and w2.

total\_count = float(sum(cfdist[w1\_w2].values()))

cfdist[w1\_w2][w3] /= total\_count

Predicting the Next words of a given Sentence

After training the n-gram model we feed it with an input sentence to predict the next word of that sentence. This is done by calculating the MLE of every word present in the corpus given a set of words present in the input sentence. The trigrams are evaluated and the prediction of the Maximum Likelihood Estimation is displayed from highest to lowest.

prediction = sorted(dict(model[prev\_words[0], prev\_words[1]]), key=lambda x: dict(model[prev\_words[0], prev\_words[1]])[x], reverse=True)

A list of possible next words is predicted in decreasing order of MLEs. For example, the input was taken as “Alice said to the” the output for the predicted words in decreasing order of MLE’s is :

['other', 'jury', 'door', 'table', 'knave', 'mock', 'gryphon', 'little', 'end', 'shore', 'beginning', 'dormouse', 'game', 'queen', 'garden', 'seaside', 'general', 'cur', 'fifth', 'puppy', 'law', 'baby', 'three', 'king', 'rosetree', 'conclusion', 'cheshire', 'duchess', 'executioner', 'croquetground', 'company', 'classic', 'whiting', 'dance', 'porpoise', 'part', 'caterpillar', 'hatter', 'head', 'tart', 'waving', 'voice', 'confused']

Now a word from this list of possibilities is chosen at random and displayed as the next word.