Project report on the implementation of n-gram models

Part 1 : Construction of the n-gram model for the given corpus and generation of random sentence with and without seed using the generated n-gram models.

Introduction

This report explains our logic for constructing unigram and bigram models from a given a training set and generating random sentences based on our constructed models. Also we test our model against seeded tokens from the given corpus to verify the functioning of random sentence generation module.

Input Data:-

We are given three different genres namely 'children' , 'crime' and 'history', where each genre consist of a collection of 3-6 text files . Few files from the given collection is used for training/constructing the model and the remaining files would be used for validation.

Output Data:-

We need to generate a set of sentences each from our unigram and bigram models. We also have to complete the sentence generation given seeded tokens using our constructed models.

NLTK tokenizer:

We have used NLTK python library to tokenize words in our training corpus.

Unigram construction logic:

Here we have created a map of all word-count pair to evaluate the frequency of each word occurring in our corpus. Next we calculate the probability of each word token in our word-count pair by dividing its frequency with the total word count.

Bigram Construction logic:

Here we created tuples for each consecutive word pair in our sentence and stored their count in map. Next we computed bigram of each word pair by dividing the count of each prior word,next word tuple by the count of the prior word as shown below:-

bigram(next_word | prior_word)= count(next word,prior word) / count(prior word)

Random unigram sentence generator without seed:

In order to generate unigram sentence without a seed, we created a helper function which will generate the next expected token to append in our unigram sentence. The helper function first creates a cumulative probability distribution of all the tokens in our unigram model. Then it uses rand() function to randomly generate a probability and return the unigram token whose probability in our distribution exceeds the generated random probability value. This helper function is repeatedly called to generate sentence till an end token is encountered or the sentence size exceeds the accepted value.

Random bigram sentence generator without seed:

In order to generate bigram sentences without a seed, we created a helper function which will generate the next expected token to append in our unigram sentence. To start, the helper function assumes the seed as the highest frequency token in our unigram model. Next it generate a list of all possible tokens that can succeed our previous token and create a their cumulative probability distribution. Now it use the rand() function to generate a random probability value and fetches the next expected token whose probability in our distribution exceed the generated random probability. This helper function is repeatedly called to generate sentence till an end token is encountered or the sentence size exceeds the accepted value.

Extending the bigram sentence generator for seed value

Given a seed token or string we proceed on the similar lines of bigram random sentence generator without the seed with the only difference here is that the initial token for the sentence generator is the last word of the seeded text while in the without seed sentence generator made use of the unigram model to initially seed in the token for bigram model.

Revisions or fine tuning that are required in the model construction

Sentence start and end marker for better predictions and better tokenization of words using nltk.