Text Generation using Markov Model

(A Suggestion-Based System)

INTRODUCTION:

Word prediction is technology that anticipates the correct word or phrase that a user wants to write after only a few characters are typed. Usually, there is a box or toolbar with several suggestions, and a click or tap on the desired choice will insert it into the text.

More advanced word predictors provide a preview option by reading the choices aloud with text-to-speech technology. For all users, word prediction can speedup the writing process and help with word recall and correct spelling.

For people with dyslexia, especially students who are still working to improve their language skills, it can act as a bridge between remediation and accommodation.

So, what is Markov property? In a process wherein the next state depends only on the current state, such a process is said to follow Markov property. For example, let's say that tomorrow's weather depends only on today's weather or today's stock price depends only on yesterday's stock price, then such processes are said to exhibit Markov property. Mathematically speaking, the conditional probability distribution of the next state depends on the current state and not the past states. That is *s(t)* depends only on *s(t-1)*, where *s(t)* is the state at time *t*. This is what is called as the first-order Markov model.

Markov Chains work by generating words based on recombination of elements of history of known sentences to generate meaningful sentences at the end.

DATA AND DESCRIPTIONS:

Our project ‘Text generation’ is based upon Markov model. We are using the power of markov model in our code to generate next word as suggestion .We will be getting most frequently used words as suggestions which will in turn help the user to recall words and will help him/her in writing quickly.

For this purpose, we have used a text file to train our data model and using that sample text, we are creating dictionaries which will suggest words according to probability of occurrences of words, i.e , words which are used more are suggested in the output.

Our data file is in plain text format.

This data file is saved in a .txt file and is used for training the model. A dictionary is made considering current word, next word and frequency of each combination of these two words from the whole text file.

METHODOLOGIES:

1. CREATING DICTIONARY FROM SAMPLE TEXT

- Store all the words in 'nextdict' dictionary
- Every word will be the key and value will be another dictionary
- Every word inside the dictionary will also contain a dictionary having the key value pairs for the words coming next to it in the sample text with value being every word's count.
- if a word is not in dictionary then we are adding that word in the dictionary and adding the word next to it and its count equal to 1 as a key value pair.
- if the word already in the dictionary then we are checking if the word next to it in the sample text is there in its value's dictionary or not. If it's there then we just increment its count by 1 and if its not there then we just add that word and keep its value as 1.

2. TAKING THE INPUT

- We will be taking input continuously from user and will concatenate all of them to return a sentence in the end.
- We take the input from user and check if its there in our dictionary or not.
- If its there in our dictionary then we take the top three suggestions from the dictionary for the word entered word and display it to user with their probabilities.
- If its not in our dictionary then we add that word in our dictionary with no word in its value.

3. BUILDING DICTIONARY FOR USER DEFINED WORDS

- If the entered word is in the dictionary of the previously entered word then we increment the count of the entered word in the dictionary of the previous word.
- If the entered word is not in the dictionary of the previously entered word then we add the entered word in the dictionary of the previously entered with count 1.

4. TERMINATING THE SENTENCE

- To print the sentence entered by user, user has to enter '.'(fullstop).

CALCULATIONS:

As we treat every word as a state and predict the next word based on the previous state. Consider the three simple sentences -

1. I like Photography.
2. I like Science.
3. I love Mathematics.

All the unique words -‘I’, ‘like’, ‘love’, ‘Photography’, ‘Science’ and ‘Mathematics’ could form the different states. The probability distribution is all about determining the probability of transition from one word to another. It is clear that first word always starts out with the word ‘I’. So there is a 100% chance that the first word of the sentence will be ‘I’. For the second state, we have to choose between the words ‘like’ and ‘love’. Probability distribution now is all about determining the probability that the next word will be ‘like’ or ‘love’ given that the previous word is ‘I’. For our example, we can see that the word ‘like’ appears in 2 of the 3 sentences after ‘I’ whereas the word ‘love’ appears only once. Hence there is approximately 67% (2/3) probability that ‘like’ will succeed after ‘I’ and 33% (1/3) probability for ‘love’.

Similarly, there is 50–50 chance for ‘Science’ and ‘fruits’ to succeed ‘like’. And ‘love’ will always be followed by ‘Mathematics’ in our case.

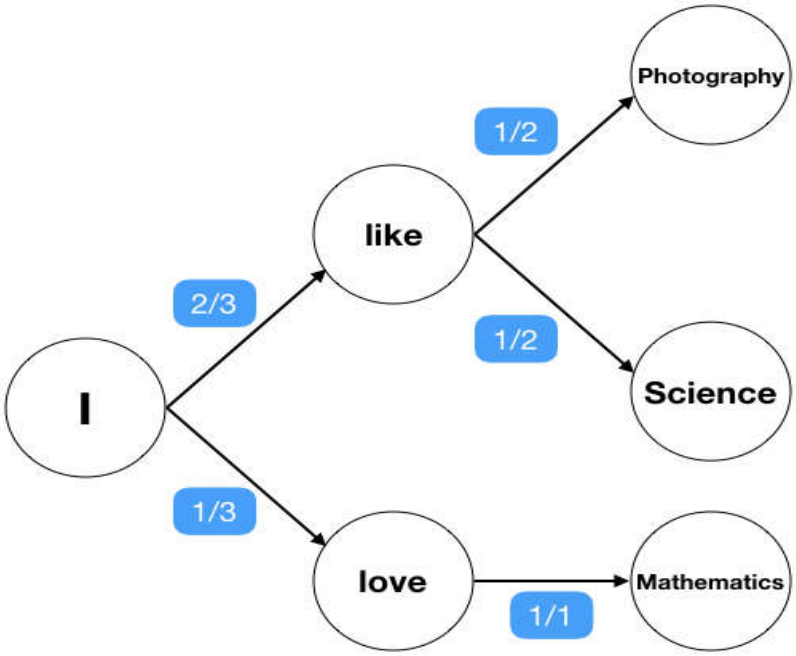

Fig 1. State transition diagram for our sample data

Representing the above work Mathematically as conditional probabilities -

P(like | I) = 0.67 P(love | I) = 0.33

P(fruits | like) = P(Science | like) = 0.5 P(Mathematics | love) = 1

This is how we build a probability distribution from a sample data.

ANALYSIS AND CONCLUSIONS:

As we can notice, Markov models do provide decent results.. It is advisable to try Markov models before jumping into much more complex models such as recurrent neural networks (RNN) or LSTM . It would be much more interesting to see how the combination of Markov models and LSTM would play out together.