Understanding Word N-grams and N-gram Probability in Natural Language Processing

N-grams

- An N-gram means a sequence of N words.
- It is about predicting the n-th word from `n-1’ words
- So for example, “Medium blog” is a 2-gram (a bigram)
- “A Medium blog post” is a 4-gram
- “Write on Medium” is a 3-gram (trigram).
- we still have to look at the probability used with n-grams, which is quite interesting.

Why N-gram though?

- Before we move on to the probability stuff, let’s answer this question first.
- Why is it that we need to learn n-gram and the related probability?
- In Natural Language Processing, n-grams are used for a variety of things.
- Some examples include auto completion of sentences (such as the one we see in Gmail these days)
- Auto spell check (yes, we can do that as well),
- We can check for grammar in a given sentence.
- We’ll see some examples of this later in the post when we talk about assigning probabilities to n-grams.

N-gram Probabilities

- Let’s take the example of a sentence completion system.
- This system suggests words which could be used next in a given sentence.
- Suppose I give the system the sentence “Thank you so much for your” and expect the system to predict what the next word will be.
- Now you and me both know that the next word is “help” with a very high probability.
- But how will the system know that?

N-gram Probabilities

- One important thing to note here is that, as for any other artificial intelligence or machine learning model, we need to train the model with a huge corpus of data.
- Once we do that, the system, or the NLP model will have a pretty good idea of the “probability” of the occurrence of a word after a certain word.
- So hoping that we have trained our model with a huge corpus of data, we’ll assume that the model gave us the correct answer.
- When we’re building an NLP model for predicting words in a sentence, the probability of the occurrence of a word in a sequence of words is what matters. And how do we measure that?

N-gram Probabilities

- Let’s say we’re working with a bigram model here, and we have the following sentences as the training corpus:

- Thank you so much for your help.
- I really appreciate your help.
- Excuse me, do you know what time it is?
- I’m really sorry for not inviting you.
- I really like your watch.

N-gram Probabilities

- Let’s suppose that after training our model with this data, I want to write the sentence “I really like your garden.”
- Now this is a bigram model, the model will learn the occurrence of every two words, to determine the probability of a word occurring after a certain word.
- For example, from the 2nd, 4th, and the 5th sentence in the example above, we know that after the word “really” we can see either the word “appreciate”, “sorry”, or the word “like” occurs.
- So the model will calculate the probability of each of these sequences.

N-gram Probabilities

- Suppose we're calculating the probability of word "w1" occurring after the word "w2," then the formula for this is as follows:

  count(w2 w1) / count(w2)

- which is the number of times the words occurs in the required sequence, divided by the number of the times the word before the expected word occurs in the corpus.
- From our example sentences, let's calculate the probability of the word "like" occurring after the word "really":

count(really like) / count(really)

= 1 / 3

= 0.33

N-gram Probabilities

- Similarly, for the other two possibilities:

count(really appreciate) / count(really)

= 1 / 3

= 0.33

count(really sorry) / count(really)

= 1 / 3

= 0.33

- So when I type the phrase “I really,” and expect the model to suggest the next word, it’ll get the right answer only once out of three times, because the probability of the correct answer is only 1/3.

N-gram Probabilities

- As an another example, if my input sentence to the model is “Thank you for inviting,” and I expect the model to suggest the next word, it’s going to give me the word “you,” because of the example sentence 4.
- That’s the only example the model knows.
- As you can imagine, if we give the model a bigger corpus (or a bigger dataset) to train on, the predictions will improve a lot.
- Similarly, we’re only using a bigram here. We can use a trigram or even a 4-gram to improve the model’s understanding of the probabilities.

N-gram Probabilities

- Using these n-grams and the probabilities of the occurrences of certain words in certain sequences could improve the predictions of auto completion systems.
- Similarly, we use can NLP and n-grams to train voice-based personal assistant bots. For example, using a 3-gram or trigram training model, a bot will be able to understand the difference between sentences such as “what’s the temperature?” and “set the temperature.”