CIS5300 - Speech and Language Processing - Chapter 3 Notes

Jonathon Delemos - Chris Callison Burch

July 6, 2025

0.1 Abstract

A **LMM** language model is a machine learning model that predicts upcoming words. These systems can create probabilties for not only the next word, but also for entire sentences. We can accurately predict an entire sentence or paragraph in advance. LM's can also use **augemntative and alternative communication** AAC. People can use these systems if they are physically unable to speak, sign, or type. Word prediction is also central to NLP for **Large Language Models**. These LLM's work by training them to predict words. An **n-gram** is a sequence of n words. Large Language Models based off **transformers** will be explained in *Chapter 9*.

0.2 3.1 - N-Grams

Suppose we want to compute the probability of the next word occurring:

P(blue | The water of Walden pond is so beautifully)

More formally, it's the same conditional independence formula we have seen in Chapter 5.

$$= \prod_{k=1}^{n} P(w_k \mid w_{k|k-1})$$

0.3 3.1.1 - The Markov Assumption

The assumption that the probability of a word depends only on the previous word is called a **Markov** assumption.

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k \mid w_{k-1})$$

0.4 3.1.2 - How to Estimate Probabilities

We estimate bigrams and n-grams by **maximum likelihood estimations** or MLE. Let's work through an example:

$< s >$ I am Sam $< s >$
$< s >$ Sam I am $< s >$
$< s >$ I do not like green eggs and ham $< s >$

Here is an example of the probability of given I.

$$P(I \mid < s >) = \frac{2}{3} = .67$$

$$P(Sam \mid < s >) = \frac{1}{3} = .33$$

Notice here, given the beginning of a sentence, there is a .67 probability I will show up.

The variables appearing in the count represent how often those two words occur in that order. In practice, language models are gigantic. This can lead to difficulty when processing the information. Instead, we use log properties to calculate their likelihood. In order to evaluate any machine learning model, we need to have at least three distinct data sets: the training set, the development set, and the test set. To find how uncertain the model is we can calculate the **perplexity**. *We just have to remember that perplexity has an inverse relationship with probability.*

0.5 3.3.1 - Perplexity as Weighted Average Branching Factor

The branching factor of a language is the number of possible next words that can follow any word.

Let's work through an example:

$$\text{perplexity}_A(T) = P_A(\text{red red red red blue})^{\frac{-1}{5}}$$

$$((\frac{1}{3})^5)^{\frac{-1}{5}}$$

$$(\frac{1}{3}^{-1}) = 3$$

Three would be the branching factor here. It is based off the number of available words in the next word bank.

P(red) = 0.8 P(green) = 0.1 P(blue) = 0.1

$$\text{perplexity}_A(T) = P_A(\text{red red red red blue})^{\frac{-1}{5}}$$

$$= 0.04096^{-\frac{1}{5}}$$

In later text we discuss *laplace smoothing* which involves adding a 1 to the numerator. This can prevent a potential zero in the conditional independence probaility calculations. An optimization to the laplace smoothing is k smoothing. The most common way to use this n-gram hierarchy is called **interpolation**. This means mixing together the unigram, bigram, and trigram probabilities.

$$\lambda_1 P(w_n)$$

$$\lambda_2 P(w_n \mid w_{(n-1)})$$

$$\lambda_3 P(w_n \mid w_{(n-2)} w_{(n-1)})$$

In the interpolation, the lambda values must sum to one. Conisder them instance weights. An alternate approach is to use **backoff**. If we arrive at a position with a zero count, we can simply backtrack (n-1) until we reach a history with some counts.

0.6 Summary

- Language models offer a way to assign a probability to a sentence or other sequence of words or tokens, and to predict a word or token from preceding words or tokens.
- N-grams are perhaps the simplest kind of language model. They are Markov models that estimate words from a fixed window of previous words. N-gram models can be trained by counting in a training corpus and normalizing the counts (the maximum likelihood estimate).

- N-gram language models can be evaluated on a test set using perplexity.
- The perplexity of a test set according to a language model is a function of the probability of the test set: the inverse test set probability according to the model, normalized by the length.
- Sampling from a language model means to generate some sentences, choos- ing each sentence according to its likelihood as defined by the model
- Smoothing algorithms provide a way to estimate probabilities for events that were unseen in training. Commonly used smoothing algorithms for n-grams include add-1 smoothing, or rely on lower-order n-gram counts through inter- polation.

0.7 Questions

I've seen some of these algorithms before: smoothing, n-gram probability calculations, sampling, cross fold validation, etc. I'm really curious how a boiler plate program could be written to employ these probability formulas.

Maybe I can write a program that will test sentiment on customer reviews?