Comparing Higher Order Markov Chains as Text Prediction Models

Ryan Hill
University of California, San Diego
rjhill@ucsd.edu

Abstract

Simple Markov chains are commonly utilized for tasks involving text prediction, such as that seen in mobile phone keyboards. The quality of these results can be improved by utilizing higher-order Markov chains, or potentially further optimized with variable-order Markov (VOM) models.

The most common default for Markov chain text prediction seems to be a state size of three, and my testing verified that third order chains yield a significantly lower perplexity than first or second order states while still keeping a more manageable state space.

1 Introduction

Text prediction has become a very common tool in a smartphone user's everyday life. Until voice interaction becomes stable and efficient enough, the main method for inputting information into an application is by typing with a virtual on-screen keyboard.

On mobile phones, this can be a huge challenge and can take large amounts of time. Unlike physical keyboards where many users have speeds of over 100 words per minute (WPM,) smartphone users type on average 27.14 WPM[1]. To counter typos and long typing times, most smartphone devices have touch input keyboards that offer suggestions for next possible words.

Text prediction is also very common for internet bots. Many parody Twitter accounts automatically post tweets that are trained to resemble the language of a specific person they are satirzing. Most of the bot's authors share information about their code online, such as "Automatic Donald Trump," demonstrating how their techniques for artificial text can produce language that real people engage with.

1.1 High Level Question

In order to address the completion scenarios at a high level, we should address the question:

What is the most likely next word given previous words?

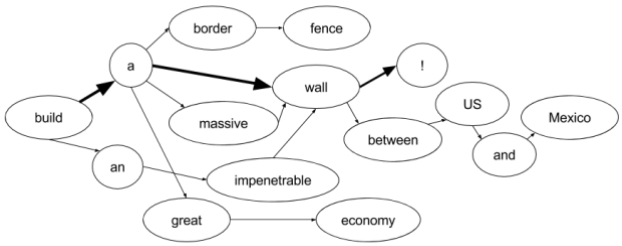
This situation can be answered without a memory of the entire history about how the text progressed to the current point in time. That is, the next word only depends on the current situation, and doesn't directly depend on language found in other paragraphs or chapters. This makes such a problem well suited for the technique we explored in class, Markov Decision Processes.

1.2 Basic Markov Chains

In class, our assignment on Markov Decision Processes was founded on the concept of the Markov property, which explains that "conditional probability distribution of future states of the process depends only upon the present state." [2]

For the Black Jack assignment, scores of the player's current hand defined each state. The value of your hand was decided by your chance of winning given what the next likely card would be, which was a separate probablity from past cards dealt. The probablities would be learned by training the MDP on simulations of the game being played.

[1] Via Quora (through TapTyping data).
[2] Wikipedia, Markov Property



As mentioned, language can be modeled similarly, with the liklihood of some next words being greater than others. These probablities can be trained through exposure to sample texts, just like the Black Jack MDP was trained through exposure to simulated card games.

Let's consider our question where we want to predict the next word a user wants to type before they actually begin typing the word. Using the most basic Markov chain, we should assume that the most recent word is the greatest predictor of the upcoming word. By training our process on a history of text, we could analyze what the highest frequency word is that follows our last known input. That is, for a sequence of words "a b," we record the probability that state 'a' is to be followed by state 'b,' as given by P(a | b), and record such values in a lookup table. The most probable next state is the one with the greatest recorded value in our table.

However, as one could intuitively guess, this model is not robust enough for many situations. Words like "to," "the," and "he" are far too common to give any meaningful insight into what the upcoming word might be. A person might "go to" anywhere, leaving this model with unrealistic predictions when only considering the previous state.

1.3 Higher Order States

One word is not enough to gauge the context needed to predict the next word; instead we need to consider the past several words (states.) At this point, our problem seems more appropriate for Markov chains utilizing higher-order states. When considering not only P(a | b), but P(a | bc), we have overall higher confidence of 'a' occuring, given that 'bc' must be either as likely or less likely than simply 'b' occurring.

Intuitively, this means our model will now have a two word memory, so predicting endings to phrases like "peanut butter" and "jelly" are much more likely, whereas the phrase "butter" alone does not provide enough context.

Higher order states also exponentially increase the size of our state space. To train our model on enough meaningful data, and to keep our state space at a manageable size, there should be a point of diminishing returns for what order state is appropriate for sufficient text prediction.

1.4 Variable Order Markov Models

In presenting hypotheticals, it is easy to conceive of situations where two or three words is not enough to have a meaningful next prediction. For example, "to do the" doesn't offer much evidence as to the next word, whereas "Kentucky Fried" obviously begs the word "Chicken."

In order to have very high accuracy, an optimal solution may involve implementing a Variable Order Markov (VOM) Model, in which all states are not bound to being a particular higher order. A VOM model would allow a mixture of first-order, second-order, and other n-order states, which may help reduce the state space from exponential expansion while still improving results.

This is outside of the scope of my experimentation, and is potentially not feasible for the given use case of text prediction on mobile phone keyboards, which cannot be intensive processes.

Instead, we will be experimenting to find the maximum order function that increases usefulness of the model before hitting diminishing returns. This should be apparent through measuring perplexity and seeing where the values converge.

1.5 Perplexity

Unlike in the Black Jack assignment, measuring outcomes is not trivial. Black Jack is a game where there are defined win and lose conditions, so we can gauge a model's prediction based on whether it accurately predicts a win or lose. Language is very subjective, and in casual keyboard usage, there often isn't a "right" answer to what the next word is.

In order to compare the usefulness of the various higher order models, we will be measuring the perplexity of each Markov chain generated. Perplexity is a measure of how well a probability distribution is able to predict a sample, and is often used to analyze language models in Natural Language Processing (NLP) research. Measuring perplexity helps understand if probabilities for given word combations are generally high, which would imply that on average, the model knows the likely next word. Maximizing the average certainty of knowing the next word is demonstrated through lower perplexity scores.

2 Experiment Design

In order to test what level of higher order Markov chains were best for predicting the next best word, I created Markov chains of orders one through six trained on the text *Harry Potter and the Sorcerer's Stone*. I then randomly sampled actual phrases from the original text, and calculated perplexity through comparing the Markov predictions to the actual text samples. For each state size, I ran 10,000 samples to ensure that the values were not biased.

The Python code that I wrote can be found on github here. I utilized the external libraries NumPy, Random, and

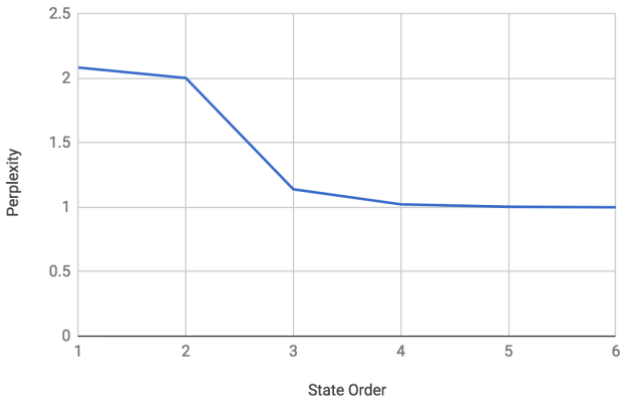
Markovify. This assisted in calculating values needed for perplexity, randomly sampling text, and for quickly generating varied state sized Markov chains to compare.

3 Results

In testing the higher order Markov chains using *Harry Potter and the Sorcerer's Stone*, I found these values:

Order	Perplexity
1	2.08470564259
2	2.00372865536
3	1.13981595897
4	1.02325856935
5	1.00429904732
6	1.00052588777

When plotting these values, it is much clearer how perplexity greatly improved with some state increases, while others had virtually no impact.



The largest change in perplexity came from increasing the state size from two to three. After that point, state size increases only minimally dropped, generally converging on the value 1.

4 Discussion

The huge gains from second to third order Markov chains largely confirms the intuition that it requires a few words of context in order to accurately predict upcoming words.

As mentioned before, there are many connecting phrases such as "to" and "the" that interfere with predictions. More unique words needed to appear somewhere in the state in order to provide context for the sentence.

Particularly in my sample training set with Harry Potter, there could very well be a lot of bias in the available language. Because the book is focused on a fictional world with many proper names and made-up terms, it could be that the characteristics of this language differ substantially from typical language used in conversation.

In addition to the raw numerical analysis, I also printed out generated statements from each of the state sizes. It's amazing how grammar rules, although not technically encoded, were generally correct for higher order states. Patterns of words that required parallelism usually appeared next to each other, so third order and above models tended to have reasonably good grammar. On an intuitive level, the models with better perplexity scores also seemed like more believable sentences.

I also know from reading the original book and printing out sample statements that some terms like "Harry Potter" are almost always together, where as connecting terms like "the," "to," "that," and others had no strong correlation. This means that while a third order model seems to generate believable text, many of the states of size three have very low probabilities and don't better the model. While "Harry Potter" is a helpful state, "Harry Potter is", "Harry Potter was", "Harry Potter needs", and other similar phrases could be cut from the Markov process without it affecting performance.

This is where a VOM should come in. Because I only tested full higher order states, it is clear that moving from a state size of two to three can cause substantial improvements, but in reality a theoretical state size of 2.6 could perform more optimally. In order to test such a theory, I would need to build out a VOM. Without this, it isn't clear if a full third order state space is needed for such improvements.

5 Conclusion

From the limited sampling of original texts to base my Markov chains on, it seems like the greatest return on state size increase comes from using third order Markov chains.

For those that would want to continue this analysis, I would recommend working with other initial texts, especially those that more accurately represent the set of commonly used English. I would also recommend experimenting with VOM models, as it is not clear if somewhere between second and third order states would offer a very similar perplexity.

References

- Markovify
- Stanford Language Modeling
- Harry Potter and the Sorcerer's Stone
- Automatic Donald Trump