Mimicking Instagram comments using Markov Chains and Natural Language Processing

Hanson Ma

November 2019

1 Overview

In probability theory, a Markov chain is a stochastic model that describes a sequence of possible following events in which the probability of each event is exclusively dependent on the state attained in the previous event. Markov chains satisfy the "Markov Property", or can also be categorized as "memory-less". A stochastic process is said to satisfy the Markov Property if the conditional probability distribution of all future states depends only on the current state, and not any state that preceded it.

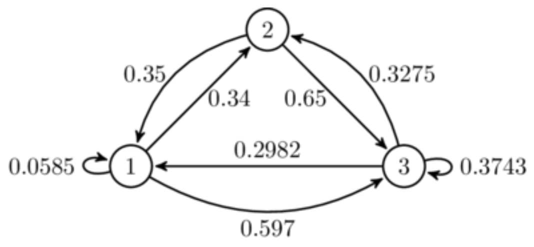

Figure 1.1 - Example Markov chain

Markov chains can be used to generate text by building "reference trees" using dictionaries. A string of text is read and reformatted to add an end-of-sentence tag, which tells the program when to stop building this specific Markov chain path. Text is then generated by randomly selecting child states, or "nodes", from the current state, and following this tree until the end-of-sentence tag is reached, at which point a sentence is generated.

This string is then indexed for every word, and a dictionary is built based on a given word, and the words that follow its every appearance in the given string. The larger the set of training data, the deeper the Markov chain, and the more varied and realistic the results will be.

2 Markov Chain Analysis

Our training data used to build the Markov chain included 1123 unique comments, each ranging from 4 to 5 words. From these comments, our algorithm built a preliminary Markov chain with 457 unique initialization nodes, meaning that there were 457 possible ways to begin a generated sentence following this Markov chain.

However, upon further analysis and testing of this particular chain it was clear that this chain was producing oddly specific sentences, far too similar to the sentences seen in the training data, down to the spelling errors and the misuse of punctuation in places. Deeper analysis reveals that this is the result of the Markov chain building very specific parent-child state pathways. In other words, there were too many states in which the current state only held one singular child state. Following the Markov process down these kinds of ”single-child-states” resulted in a sentence of very specific structure and word order.

This analysis was done by recording the number of child states each parent state had, from the start of Markov chain construction to the end of the training data file. Plotting the number of child states as a function of parent states ordered by appearance in training data, a scatter plot shows the convergence to the single-child-state value of 1.

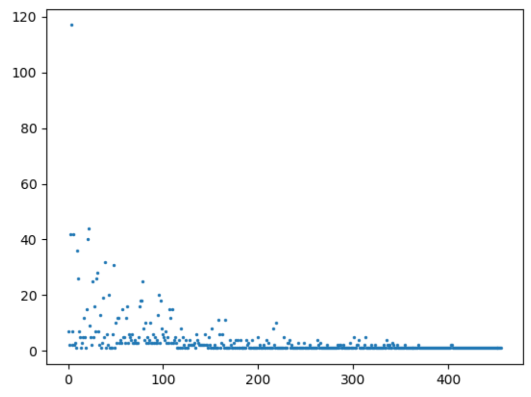

Figure 2

Applying exponential regression to this data set gives us the rough fit equation of:

$$f(x) = 102.9128 \cdot 0.98^x$$

Taking the integral of this function in the range of zero to the near convergence of $f(x) = 1$, gives us the quantification of roughly how many words were needed for the Markov chain to "understand" the language nature of these comments. Integrating this function from zero to a rough convergence estimate of $x = 200$ gives us:

$$\int_0^{200} 102.9128 \cdot 0.98^x dx = 5004.42$$

Multiplying our original number of unique training comments by a rough estimate of average number of words per comment gives us $1123 * 4.5 = 5053.5$, a slight order of magnitude higher than the estimate, backing up the conclusion that the current Markov chain is unnecessarily specific, and would need to be modified in order to construct a more general, more realistic model.

3 Markov Search Tree Pruning

In order to amend this unnecessary specification in the Markov chain, pruning was implemented in order to remove the single-child-state cases. A Markov chain dictionary was run through a parser/safe-delete algorithm in order to find all single-child-state cases, and remove not only the parent nodes, but all equivalent child nodes.

However, this presents a new issue; pruning the search tree for single-child-state cases means deleted not only the respective parent node, but all cases where that parent nodes appears as a child node. As a result of this safe-delete pruning, some other parents nodes might go from multi-child-state cases to single-child-state cases. Although the number of these cases is greatly decreased by a first-pass pruning operation, the effects of this operation leaves residual single-child-state cases. Multiple passes of pruning operations may decrease this number, but that is highly dependent on the state of the transition matrix.

On average, Markov search tree pruning removes 55% of all start cases. This new Markov chain generated much more generalized, realistic sentences containing the language patterns of the training data, but not the specifics of them.

4 Weighted Selection Library Construction/Feed-forward decision making

Despite the improved model, the constructed sentences still lacked a certain level of structure and accuracy, with respect to both grammar and language patterns. The solution to this was to introduced weighted feed-forward selection libraries in to the Markov decision making process.

Currently the Markov decision process follows the Markov chain by randomly selecting a child state from its current parent state, and iterating over that child-parent state swap until it reaches an end-of-sentence flag. The list of child states it randomly selects from are directly pulled from the Markov chain, and as such lack a certain level of stochastic sophistication.

Instead of directly pulling a random child state from the parent state, the new Markov decision process looks ahead in the Markov chain for every potential child state in the parent state. It counts how many "grand-children" states each child state has, and weights the probability of selecting a given child state accordingly.

The effect of this feed-forward logic implemented into the Markov decision process is that the model now selects chains of words that are more likely to appear one after another, instead of pure pseudo-randomly selecting child states. The greater the number of child states in a given parent state, the higher the probability that that parent state is likely to exist in real language.

Regardless, as a result of subsequent implementations to the model, search tree pruning, weighted selection libraries, and feed-forward decision making, the Markov Decision Process now constructs highly accurate and generalized results, only to improve with the addition of training data.

5 Markov State Transition Matrix

Before Weighted selection libraries, the probability distribution of all child states given a parent state was evenly distributed. With the introduction of weighted libraries, those probabilities changed, which can now be represented with a square matrix.

To represent this adjusted Markov chain in a mathematical sense, a Markov State Transition Matrix can be constructed to represent the probability distribution of child states given a parent state. Because sentence generation includes end cases, a Markov State Transition Matrix for text generation includes a final ”end case” row and column.

Individual vectors for each row are constructed by calculating the probability of a given word’s appearance in the weighted selection library, and adding that probability to its respective row and column. Because of the nature of the language in question, state transition matrices for generating these kinds of texts include a lot of zeroes, meaning that in any given parent state, the probability of a selecting an end-of-sentence tag is much more likely than selecting any child state.

Feel free to view or download our source code at:
https://github.com/hansonhqma/basicbot