DOI: 10.53469/jerp.2024.06(09).05

Explore the N-Gram Model of Adaptive Prediction Text

Brahmaleen Kaur Sidhu

Aarav Rathi, Fremont High School
sidhu360@gmail.com

Abstract: ***This program is aimed towards enabling people with speaking disabilities to participate more within their conversations. People with speaking disabilities are often forced to rely upon sign language or some form of text to speech to communicate in their day-to-day life. This often creates trouble as some people may not know sign language, and typing out every single thing you may want to say takes a lot of time and effort. This program will help these issues by creating a text-to-speech text generator. By using pattern recognition, the program will learn the person's talking styles, and be able to more fluently autofill the sentence; thereby requiring less effort and time on the user. The program uses a probabilistic n-gram model in order to predict what the user might want to say in real time. By using the user's input as training data in the future, the n-gram models can adapt to the style and tone of the user reasonably quickly.***

Keywords: speaking disabilities, communication aid, text-to-speech, pattern recognition, n-gram model

1. Introduction

People with speaking disabilities oftentimes cannot participate in a conversation the same way as someone who is abled can, due to long times of texting/translating. Jokes that needed to be inserted at specific points, or asking a clarifying question during a natural lull of the conversation are just some of the few things that are overlooked by many, but have a drastic effect on the conversation when they go unfulfilled.

How accurately can ML algorithms predict and generate texts in multiple languages after being tailored with new user input? Purely machine learning based models are usually quite tricky to train for the intricacies of human speech, both with content and grammar. Even large projects like ChatGPT utilize a human based learning system in order to efficiently develop a robust chatbot. That is why instead of using a typical ML setup, we will be reusing the user's input as training data. This saves much time building a text-generator as now a lot of the data is certified to come from real conversations. We hope to bring out the inference for the future that other chatbot/text generation models can be easily made by slotting in user entry as training data, in order to skip making an incredibly robust and grammatically correct model that can adapt to many styles all at once. The con of this methodology is that in the beginning the model has very little user data to base itself off. But by implementing a "preset" of sorts, the user can start off with an already implemented set of text based on several differing personas, giving the model a little breathing room until enough user input has been registered to swap over.

2. Literature Review

A source we took inspiration from was the "Multilingual Healthcare Chatbot Using Machine Learning" by Sagar Badlani, Tanvi Aditya, et al [1]. This research paper, while not terribly similar to what we are doing, have a very well fleshed out pre-preprocessing step for NLP preprocessing. It enunciated several key processes such as tokenization, stop word removal, stemming, etc. Data-preprocessing is a subtle thing in NLP, as there are hundreds of factors to consider and not all of them have a linear relationship with the outcome. Many only strive to improve their models, but forget that without proper data, even the best models would fail. But speaking of models, the "Neural Network Approach to Word Category Prediction for English Texts" paper by Masami Nakamura, Katsuteru Maruyama, et al [2] discusses the benefits of a n-gram model approach. A struggle with NLP is the complexities that are intrinsic with human language. It is very difficult to create a standard model that can find the patterns between all of our multifaceted communication. The paper introduces an n-gram model, a model that works heavily with the data provided to create realistic human sentences by grafting parts of data together, all at a very low computational cost. We employed this model heavily throughout our program, going so far as to use multiple variants to better account for different situations.

3. Methodology

The first step in recreating the core parts of this program is to set up an n-gram model, all the way from bi to quad grams. This can easily be done by utilizing the ngrams module from the natural language toolkit library. This library makes it very simple and easy to set up an n-gram model. From there you have to select data to run the model on. It is recommended that you have a sufficient amount of sentences in each style of writing if you choose to create multiple presets as well as ensuring the styles of speech are distinct enough. Cleaning the data is a big part, due to how these models work. The data preprocessing we did was adding all the data together while getting rid of all formatting: leaving us with just sentences. Getting rid of formatting, strange characters, and extra spaces can all be simply done by a looping mechanism. The UI does not really have an effect on the actual code that does the designated task, so whatever format is the easiest or best looking can be used. Once a sentence has been outputted and selected, in order to achieve an adaptive feature, you must add the sentence to the data pool. Due to a varying amount of factors, we decided that it is best to add a dynamic weight to the sentences that are added to the datapool (inversely related to the amount of sentences that are added) in order to see the adaptive effects much earlier.

4. Results

After running the program on several sets of data, we have come to the conclusion that while the AI can be strongly accurate in predicting and generating text, it often needs a vast amount of data in order to create organic responses a majority

of the time. Smaller datasets that were used for testing oftentimes had sentences that were 1-to-1 copies of what the n-gram model spat out, showing a clear lack in variability. While this is to be expected when a rare word is selected from the rest of the options, having sentences being copied due to a 'relatively common' (up to personal interpretation) word clearly shows a lack of data. While at its core the n-gram model is nothing more than a sophisticated copying machine, when the copying is entire sentences at a time, you have to add more data to the pool.

Table I: Breakdown of the data between types of datasets & amount of information each one contains

Dataset Name	Number of Reviews	Number of Sentences	Sentences per review
Starbucks	850	4,884	5.75
Disney Land	42,656	3,09,814	7.3
Restaurant	3,640	26,537	7.3
Fridge	10,000	40,101	4

Table I: One of the most important segments during the beginning phases of usage, due to the outputs being derived off the data

- Higher sentences per review cater to different speech styles than lower ratios do

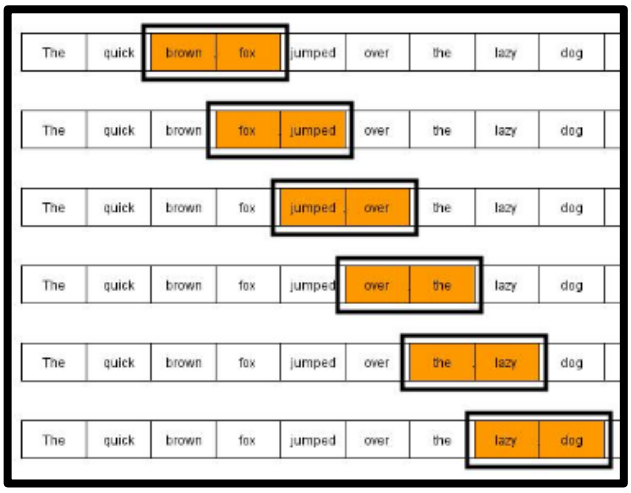
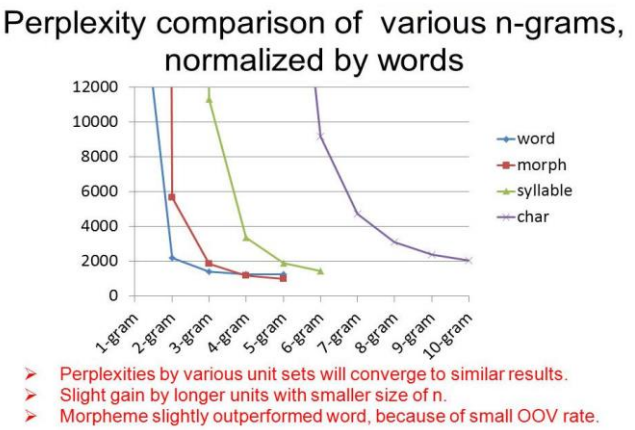


Diagram I: A visual method of how a bi-gram (2) works; what section of the sentence it considers during each iteration [3]

Diagram 1: An n-gram model takes into account the current word, and the last n-1 words when deciding what to iterate through

- Bi-gram models repeat one word each time, tri-gram models repeat two words each time, and so on
- The model uses the selected words to look for phrases in the data with the exact same selection of words, adding the words that follow the phrases in the data to a hat to randomly pick from



Graph I: A perplexity comparison of different n-gram models based on several factors [4]

Graph I: Perplexity is the inability to deal with or understand something, and this graph shows how lower "n" n-grams struggle more with certain linguistic concepts that the higher n-grams do

5. Discussion

N-gram models take a look at the last n words in the sentence when deciding what to add as the next word. Already we have noticed some differences in n-gram models (where n has changed) when attempting to mimic different styles of speech. Whereas some styles might be short and informal, others might be long, showing a need for different n-gram models. You can see some of the differences between how n-gram models work on the chart above, showing the relationship between n (in ngram) and perplexity comparisons. While larger n-gram models do tend to tend to perform better, they do need more data to function organically.

We suggest that outputs shown are not just from 1 n-gram model type, but from many differing types. These types can be chosen by going through a "hat" that has every previously selected option's model type. For example, if a user has selected 3 sentences from a bi-gram, 8 sentences from a tri-gram, and 15 sentences from a four gram, the models chosen to generate sentences will be chosen from those 26 options. While in the beginning you may need to add some artificial options, as time goes on and the program learns to adapt itself to the user, it should run more smoothly. Everyone's style of speech is different, and in order to create a text generator that matches the individual user, the program should attempt to tune most of its major parts in order to fit the user.

6. Conclusion

To answer the question of 'How accurately can ML algorithms predict and generate texts after being tailored with new user input', the answer is pretty accurate. A key is to realize the different core parts that create a person's unique speech characteristics, and to link them to different parts of the code (or create parts that represent them if none are available). By first making sure a user's style is inextricably linked with the code, we can make any auxiliary features based around it (i.e. a segment that adds language translation).

The program can then become self-sufficient by learning off future inputs to ensure it always remains up-to-date, a feature whose adaptive qualities lead to better, more tailored results. Due to this, it recommended that any who attempt to create a similar design take the time to experiment to see what settings would fit what they are looking for. Things such as data, model type, and even accounting for how data cleaning changes are all factors that can affect the outcome, and therefore are to be taken into account when matching a set of weights to a user.

Acknowledgement
Thank you for the guidance of Emily Sheetz, mentor from University of Michigan in the development of this research paper.

References

[1] Badlani, Sagar, et al. "Multilingual healthcare chatbot using machine learning." *2021 2nd International Conference for Emerging Technology (INCET)*, 2021, https://doi.org/10.1109/incet51464.2021.9456304.
[2] Nakamura, Masami, et al. "Neural network approach to word category prediction for English texts." *Proceedings of the 13th Conference on Computational Linguistics-*, 1990,
[3] Ajitesh Kumar. "N-Gram Language Models Explained with Examples." *Analytics Yogi*, 2 Feb. 2018, vitalflux.com/n-gram-language-models-explained-examples/
[4] Arvind Pdmn. "N-Gram Model." *Devopedia*, Devopedia Foundation, 1 Mar. 2023, devopedia.org/n-gram-model.