ECE 684 Final Project

Richard Fremgen, Johnny Antoun, Jaskaran Singh

17 December, 2022

---

1. Introduction

While there are many natural language processing (NLP) applications that people interface with regularly, perhaps the most common is language modeling, which can be seen in most digital applications, such as in search engines, email programs, and text messaging. Language modeling is commonly used for text generation NLP problems, where a system is trained to generate a sequence of text based on the words that have been previously entered. Text generation is utilized within many common NLP tasks such as for machine translation, auto-complete, and speech-to-text applications. Anytime one searches the web to look up a question, writes an email, or sends an iMessage, they are interfacing with some form of language modeling system that is trying to predict the next sequence of word(s) in order to yield better search results. Much of the mainstream NLP news nowadays concerns advances and iterations of the techniques utilized in Generative Pre-trained Transformer 3 (GPT-3) and Bidirectional Encoder Representations from Transformers (BERT) models that are the essential underpinnings of numerous web and mobile applications. While there are a variety of approaches to accomplish the task of text generation, the two most common use a statistical technique such as an N-gram or Hidden Markov model, in addition to building a neural network such as a bidirectional RNN or LSTM to generate text. At the core, both models are simple: assigning a probability to a sequence of words given what has already been written in the text; the computational and statistical approach of how that probability is computed and arrived at comprises the differences of such. In this paper, we apply both approaches to a real and synthetic data set, and discuss the advantages and limitations of both classes of models.

2. Generative Probabilistic Model

The most heavily used generative probabilistic model to solve the NLP problem of text generation is a Markov model, which is also referred to as the *n-gram* model. This model uses the *n-1* words in the past in order to compute the probability of the next word, w_i, occurring, which can be represented mathematically as such:

$$p(w_i|w_{i-1}, \ldots, w_1) = p(w_i|w_{i-1}, \ldots, w_{i-n+1})$$

This means that if we were given the sentence `Tomorrow at noon, he has his __`, and were using a trigram model (*n=3*), to calculate the most probable next word, we would solve for the following:

$$p(w_3|\ w_2 = \text{his},\ w_1 = \text{has}) = p(w_3|\ \text{has}\ , \text{his})$$
$$\therefore\ \hat{w}_{3,\text{MLE}} = \arg\max_{w_3}\ p(w_3|\ \text{has}\ , \text{his})$$

In order to build a generative probabilistic Markov model, we used Homework Assignment 3 as a motivating example to define a function `finish_sentence` that served as a bare-bones Markov text generator given the following inputs:

- `sentence`: list of tokens for the sentence you want to generate additional text for

- `n`: an integer corresponding to the length of n-grams used for prediction
- `corpus`: list of tokens that serve as a source for which the model is trained on
- `deterministic`: Boolean value that dictates whether a deterministic or stochastic process is used to generate text. When `deterministic = True` is set, the deterministic method of generating the next word in a sequence is utilized, meaning that the most likely token (from a probability standpoint) is selected. When `deterministic = False` is set, a stochastic text generation method is utilized, where the n-gram model assigns a conditional probability to each possible next word and then draws from this conditional distribution. To accomplish the stochastic feature of this function, the `np.random.choice()` function was used to sample from this distribution, meaning the the stochastic output of `finish_sentence()` will differ given the same input tokens over numerous iterations.

In addition to the features described above, backoff was also integrated into this function, which is used in n-gram models when the function comes across a rare token or sequence of tokens that were not in the training set. In such a scenario, the backoff feature of the `finish_sentence()` will reduce the value of `n` until a match is met from the training data (`corpus` input). For example, if we input `sentence = ['tomorrow', 'at', 'noon', 'he', 'has', 'his']` and `n=5` into our function, initially we will try the find the word following `his` by solving $P(next\ \ word|\ noon,\ he,\ has,\ his)$. However, if the phrase `['noon', 'he', 'has', 'his']` never appeared in our training corpus, then backoff will occur where `n=5` is reduced to `n=4`, meaning that we then be solving for the conditional probability $P(next\ \ word|\ he,\ has,\ his)$. Such a process will repeat until a match is found, or the n-gram model for that missing word is reduced to a unigram model. Once a word is found, the value of `n` will return to the original n value that the user inputted. Other features of the `finish_sentence()` function include the following:

- **Two Tokens Equally Probable**: when `deterministic = True` there is a chance that two (or more) tokens will be equally probable as being the next word in the sequence. In such a case, the word that appears first in the corpus will be returned, meaning that out of the *tied* words, the word with the smallest index in the inputted `corpus` will be selected. (Motivation provided from HW 3)
- **What Happens when n-1 > len(sentence)**: If the user were to input an `n-1` value greater than the actual length of the input `sentence`, then the function will backoff until an adequate `n` can be used. For example, if the user inputs `sentence = ['she', 'was', 'not']` and `n = 5`, the function will automatically backoff to `n=4` for when attempting to fill in the fourth word in the sequence, but will return to `n = 5` (the value the user set), once the sentence list is long enough to support the requested `n` value.

3. Discriminative Neural Network

In addition to using Statistical Language Models such as the N-gram Model or Hidden Markov Models (HMM) which fall under generative approaches, a common approach to text generation is implement a Neural Network model. Borrowing motivation from the ideas and code presented in *Build a Natural Language Generation (NLG) System using PyTorch* (see `References` section for url), we opted to use an LSTM, a special case of a Recurrent Neural Network to generate text. Using an approach such as an N-gram Model has its limitations. If we are using a bigram model, we are not incorporating any of the context from a word that is five words away. For example if we have sentences such as: `Richard went to class. Tomorrow at noon, he has his __`, a bi-gram model would not incorporate information from the word `class` to generate logical solutions such as `exam`, `text`.

To capture such unbounded dependencies among the tokens of a sequence, we can use an LSTM-based model. The model we decide to use consists of four layers (in the following layering order): embedding layer, LSTM layer (allows remembering/forgetting mechanism), dropout layer (reduces overfitting), and a fully-connected layer (to produce a vector of length vocab size). The inputs and targets given to the Neural Net overlap in one word. For example an input would be `He is not` and its respective target would be `is not happy`. The length of the input and the target is a hyper parameter we can tune but we found through training that the best performance was with inputs and targets of length two (included are Jupyter notebooks for lengths of inputs and targets 2 and 5 along with their results). We transform words from strings to numbers before imputing into Neural Network. We opt to use a batch size of 32, meaning that we are giving the network an input of size

32 (batch size) by 2 (length of input and target). The final output is of dimensions 64 by vocab size. We can interpret the output as representing probabilities of each word in the vocabulary. In terms of a loss function, we opt to use cross-entropy loss (classification problem) and the Adam optimizer; both model architectures were run for 20 epochs. The Neural Net is used inside a sampling function that produces text given the size of the text generation, starting string, and Neural Network architecture. We use the output of the predict function to evaluate performance against the N-gram Model discussed in `Section 2`.

4. Real Data Training and Application

4.1 Real Data Introduction

In order to evaluate the performance of the Markov model and LSTM network, both models had to be trained on the same data set. To accomplish this feat, a sample of the Carnegie Mellon University (CMU) Movie Summary Corpus was used. The CMU Movie Summary Corpus is comprised of 42,306 movie plot summaries that were extracted from Wikipedia and Freebase by David Bamman and colleagues at CMU. Due to computational and time constraints, 500 movie summaries were sampled from this data set, each of which were roughly the size of an average English language paragraph. After data pre-processing occurred and all 500 movie summaries were aggregated into a single list, this resulted in a corpus of approximately 157,000 tokens that was then used to train both the generative and discriminative models in the subsequent sections. As such, listed below are the results from five different input sequences that were tested on various configurations for both the Markov model and LSTM network, which where trained on the Movie Summary Corpus described above. The `N` column represents the type of *n-gram* model that was used for each method (e.g. $N = 2$ means the model was a bigram), and the `Deterministic` column corresponds to whether or not a deterministic (`Deterministic = TRUE`) or stochastic (`Deterministic = FALSE`) process was used. Additionally, the `Output` column depicts the text each model generated given the input sequence, as all model configurations were limited to generate ten words (including the input sequence).

4.2 Example 1

Table 1: Input Sequence: `the girl`

Method	N	Deterministic	Output
Markov	2	TRUE	the girl who is a new york city and the
Markov	3	TRUE	the girl who is a good thing but the other
Markov	5	TRUE	the girl who 'd appeared and disappeared so mysteriously during
Markov	2	FALSE	the girl should stay in a surprise birthday before he
Markov	3	FALSE	the girl finally accepts and starts insulting ray by talking
Markov	5	FALSE	the girl who saved his life eric rushes to kiss
LSTM	2	TRUE	the girl sunita lie deportation future beck patrolman stumble lie
LSTM	5	TRUE	the girl mystery peasants' cowboys mystery assimilates dip issue attendants

4.3 Example 2

Table 2: Input Sequence: `the man told`

Method	N	Deterministic	Output
Markov	2	TRUE	the man told that he is a new york city
Markov	3	TRUE	the man told that the man who has been in
Markov	5	TRUE	the man told that the jewellery was given to him
Markov	2	FALSE	the man told that jeff poindexter a medical clearance to
Markov	3	FALSE	the man told the chairman concludes the hearing and praetorius

Method	N	Deterministic	Output
Markov	5	FALSE	the man told to begin with words on why he
LSTM	2	TRUE	the man told bruce lie brazilian sunita lie deportation sunita
LSTM	5	TRUE	the man told emil's swordsman glacier upright mystery assimilates dip

4.4 Example 3

Table 3: Input Sequence: `as soon the`

Method	N	Deterministic	Output
Markov	2	TRUE	as soon the film ends with the film ends with
Markov	3	TRUE	as soon the whole thing but the other hand is
Markov	5	TRUE	as soon the whole family is being neglected by the
Markov	2	FALSE	as soon the movie has managed to armsized creatures in
Markov	3	FALSE	as soon the whole thing screams in anguish without thinking
Markov	5	FALSE	as soon the whole family is being neglected by the
LSTM	2	TRUE	as soon the spotted sunita lie protects stumble lie brazilian
LSTM	5	TRUE	as soon the mystery chan's swordsman comedian wrecks dip issue

4.5 Example 4

Table 4: Input Sequence: `i have`

Method	N	Deterministic	Output
Markov	2	TRUE	i have been a new york city and the film
Markov	3	TRUE	i have been a martial arts competition the international kindergarten
Markov	5	TRUE	i have been waiting ten years for this he says
Markov	2	FALSE	i have transferred them dad ali lectures frida he even
Markov	3	FALSE	i have been located by the th century castle originally
Markov	5	FALSE	i have been waiting ten years for this he says
LSTM	2	TRUE	i have bruce lie shah's stumble lie brazilian sunita neverending
LSTM	5	TRUE	i have leprechauns dip issue bums cowboys mystery detects cowboys

4.6 Example 5

Table 5: Input Sequence: `a car`

Method	N	Deterministic	Output
Markov	2	TRUE	a car and the film ends with the film ends
Markov	3	TRUE	a car accident with no apparent disapproval from the city
Markov	5	TRUE	a car accident with no child to tie them together
Markov	2	FALSE	a car into the mountains while training in his suspicions
Markov	3	FALSE	a car chase that follows the fight back and forth
Markov	5	FALSE	a car containing a shipment of gold bullion robert 's
LSTM	2	TRUE	a car lie brazilian sunita lie brazilian sunita neverending stumble
LSTM	5	TRUE	a car mystery chan's dip issue man' recent swordsman mystery

4.7 Real Data Text Generation Results

Looking at the results of the text generation for the five input sequences above, we can see that based on visual inspection, the various Markov models outperformed the two different LSTM model configurations. Such can be seen based on the observation that the Markov models (especially in cases where $N = 3$) produced sequences that are more discernible and correspond to phrases (or fragments of phrases) that would be common for the English language. Ideally we would have expected the LSTM model to outperform the Markov model, since the Markov model assumes that the data comes from a generative process, which when using real data is not always found to be true. However, one explanation for this occurrence has to do with the constraints imposed on the corpus, where due to computational cost, we decided to use only a subset of the CMU Movie Summary Corpus, rather than use the entire corpus, which would have taken considerable more time and computational power to train on. The strength of any neural network is determined large in part by the amount of data utilized during training, as the accuracy of deep learning systems drastically increase with the more data it is provided. Another explanation for the Markov model generating more discernible text is the type of corpus used for this project. Taking a closer look at the corpus, it is evident that a good portion of the text used in the training is comprised of specific nouns such as locations or people's names used in the film. While such is conducive to properly summarize a film into a paragraph, using such a corpus did mean that there were a large number of infrequent words (e.g. specific names or people or place) that impacted the lucidness of the model output. A final explanation for the Markov model outperforming the LSTM has to do with that fact that perhaps the LSTM used was not complicated enough from a model architecture standpoint. Future work on this topic should be focused on enhancing the neural network complexity and increasing the number of hidden layers to see if such will improve the text generated.

5. Synthetic Data Training and Application

5.1 Synthetic Data Introduction

In addition to training the two different classes of models on real data, both approaches were also tested on synthetic data that was generated according to the generative Markov model described in `Section 2`. In order to accomplish this, the Markov model trained on the movie corpus data was given 500 input sequences, which it then used to generate 500 artificial or synthetic pieces of text. This *synthetic* data corpus was then used in the subsequent sections below to retrain a Markov and LSTM model and then generate new pieces of text, given the same 5 input sequences from `Section 4`. Such synthetic data generation process was executed both from a deterministic and stochastic perspective (by modifying the value of `deterministic` in the `finish_sentence()` function) as both synthetic data sets were used for training. The `Syn Data Type` column below indicates which synthetic data set, the deterministic (`Det`) or stochastic (`Non-Det`) corpus, was used to generate the output below.

5.2 Example 1

Table 6: Input Sequence: `the girl`

Method	N	Syn Data Type	Output
Markov	2	Det	the girl attending a new 'master' who is a
Markov	3	Det	the girl attending a college a long way from home
Markov	5	Det	the girl attending a college a long way from home
Markov	2	Non-Det	the girl attending a young lady beatrice russo who is
Markov	3	Non-Det	the girl 's in the back where she recognizes her
Markov	5	Non-Det	the girl 's in the spring unfortunately evil magician murgatroyd
LSTM	2	Det	the girl named leaving that is her arrested of tells
LSTM	2	Non-Det	the girl of the film of her be ' ands

5.3 Example 2

Table 7: Input Sequence: `the man told`

Method	N	Syn Data Type	Output
Markov	2	Det	the man told to the three children for the three
Markov	3	Det	the man told to wait back holds his games refuse
Markov	5	Det	the man told to wait back holds his games refuse
Markov	2	Non-Det	the man told to the film also shocked as the
Markov	3	Non-Det	the man told to wait back holds his games refuse
Markov	5	Non-Det	the man told to wait back holds his games refuse
LSTM	2	Det	the man told the apartment green guerrero becomes first washington
LSTM	2	Non-Det	the man told and the three was was was was

5.4 Example 3

Table 8: Input Sequence: `as soon the`

Method	N	Syn Data Type	Output
Markov	2	Det	as soon the three children for the three children for
Markov	3	Det	as soon the three men conducted a guerrilla war against
Markov	5	Det	as soon the three men conducted a guerrilla war against
Markov	2	Non-Det	as soon the film also shocked as the film also
Markov	3	Non-Det	as soon the film also chika puts her furs back
Markov	5	Non-Det	as soon the film also chika puts her furs back
LSTM	2	Det	as soon the film i i i leaving to attract
LSTM	2	Non-Det	as soon the woman ' who her three more was

5.5 Example 4

Table 9: Input Sequence: `i have`

Method	N	Syn Data Type	Output
Markov	2	Det	i have to the three children for the three children
Markov	3	Det	i have to murdered girls are buried in anand 's
Markov	5	Det	i have to murdered girls are buried in anand 's
Markov	2	Non-Det	i have to the film also shocked as the film
Markov	3	Non-Det	i have to murdered girls are buried in anand 's
Markov	5	Non-Det	i have to murdered girls are buried in anand 's
LSTM	2	Det	i have and avoid for the apartment who is first
LSTM	2	Non-Det	i have and the three was was was was was

5.6 Example 5

Table 10: Input Sequence: `a car`

Method	N	Syn Data Type	Output
Markov	2	Det	a car around the three children for the three children

Method	N	Syn Data Type	Output
Markov	3	Det	a car around boys before the festival the pool is
Markov	5	Det	a car around boys before the festival the pool is
Markov	2	Non-Det	a car around boys before the film also shocked as
Markov	3	Non-Det	a car compelled to tell her he a raft they
Markov	5	Non-Det	a car compelled to tell anna of jacob now the
LSTM	2	Det	a car i i i i contra rebels becomes guerrero
LSTM	2	Non-Det	a car of her young children who her three

5.7 Synthetic Data Text Generation Results

Similar to the results in `Section 4`, we can see that the Markov model generated more discernible text upon visual inspection when compared to the LSTM network. Such results are expected and can be attributed to the fact that the data used to train both models was generated according to a certain distribution that was specified by the Markov generative model. The Markov model in turn generated new text under the same probability distribution assumption, meaning that it is reasonable for the Markov model to produce more legible text. Since the data generation and model calibration assume the same data generative process, we can see in the results above, that the Markov model produced output that more closely resembled traditional English verbiage. On the contrast the LSTM model generated sentences that appear to be randomly selected words, where one word is repeated numerous times in certain instances.

6. Comparison

6.1 Generative Model

The most salient feature of generative models is that they are interpretable under the probabilistic framework. The generative models account for the uncertainty (variance) in the data under the specified probabilistic model which captures prior beliefs about the data generation process. This lends interpretability and intuition to the model. Another advantage of generative models is that they require less computational resources and less data to train as compared to discriminative models. The reduced time complexity of generative models makes them suitable for real time applications. However, the results/predictions from generative models are optimal (minimum Bayes error) if the real world data actually follows the assumed generative model. This inherent assumption about the data generating process makes it vulnerable to generative model misspecification.

6.2 Discriminative Models

The discriminative models are currently State of the Art models for various machine learning tasks, especially for images and languages. They have the ability to learn complex non-linear functions and thus have made remarkable progress for prediction related tasks, owing to their superlative predictive accuracy. However, discriminative models can often involve large model parameters which entail large computational resources, more time and large datasets to train these models. Further, the discriminative models are less interpretable as compared to the generative models.

7. References

- ECE-684 Notes and TA Office Hours
- ECE-684 HW 3 (Fremgen)
- ECE-684 HW 3 (Antoun)
- ECE-684 HW 3 (Singh)
- *Build a Natural Language Generation (NLG) System using PyTorch* - Code for LSTM Model (Section 3) https://www.analyticsvidhya.com/blog/2020/08/build-a-natural-language-generation-nlg-system-using-pytorch/

- *Text Generation using PyTorch LSTM Networks* https://coderzcolumn.com/tutorials/artificial-intelligence/text-generation-using-pytorch-lstm-networks-and-character-embeddings
- *CMU Movie Summary Corpus* - http://www.cs.cmu.edu/~ark/personas/
- *Learning Latent Personas of Film Characters* - David Bamman, Brendan O'Connor, Noah A. Smith
- *Speech and Language Processing* - Daniel Jurafsky & James H. Martin

---