Language Modelling of Gutenberg corpus

Saurabh Annadate
Master of Science in Analytics
Northwestern University
2145 Sheridan Rd, Evanston, IL 60208

Abstract

In this work, we explore three model architectures which include statistical N-gram models and two Recurrent Neural Network architectures for large scale language modelling. A subset of the Gutenberg eBooks corpora was used as the corpus on which the models were fit. Several hyperparameters were tested and the performance metrics as well as the training time for the models were recorded and analyzed. Due to the difference in the architectures, the model performance metrics cannot directly be compared amongst the different architectures.

The code to reproduce the findings can be found at https://github.com/saurabhannadate93/Text-Analytics-Language-Modeling. The repository contains code to fetch the training corpora, clean the data, fit the models as well as expose the best performing models for all three architectures via REST APIs.

1 Introduction

Language modeling (LM) is the task of modeling the probability distribution of the sequence of words/characters in a corpus. It plays a key role in several NLP tasks such as speech recognition [1][2] and text summarization [3][4]. Therefore, improving the LM performance is very valuable as it will improve the underlying metrics of the downstream tasks. Language Models also find their application in seq-to-seq models which are used in machine translation [5][6] and video generation [7].

There are two primary types of language models - Statistical Models which are the traditional models and Deep Learning models which are more recent. The statistical models use techniques like N-grams, hidden markov models, and certain linguistic rules to learn the probability distribution of words. The Deep Learning Neural network models use neural networks to learn the language structure to predict the next word/character given an input context.

In our work, we train the models on a subset of the Gutenberg eBooks corpora which can be considered as a medium sized dataset for the n-gram models but a very large size dataset for the neural network models. This dataset has not been used for language modelling before. For one of our RNN models, we experiment with a different unseen architecture which models the embedding of the next word rather than the word itself.

2 Related Work

In this section, we discuss previous work relevant to the approaches discussed in the paper.

Language Models
The aim of a language model is to learn a probability distribution over sequences of symbols pertaining to a language. A lot of work has been done on both parametric as well as count-based approaches (N-gram models). In the past five years, a lot of progress has also been made to train deep learning neural network models to learn the language distributions with good accuracy, especially Recurrent Neural Networks and its variants which can learn and retain long term dependencies.

N-gram Models
An N-gram model predicts the probability of a given N-gram within any sequence of words in a language. If we have a good N-gram model, we can predict p(w/h) - what is the probability of seeing

the word w given a history of previous words h - where the history contains (n-1) words. The way the model achieves this is that it counts how many times the word w appeared in the context h, and normalizes by all observations of h. Although N-gram models are theoretically sound, they have a few limitations [8]:

1. Many histories h are similar, but n-grams assume exact match of h
2. Practically, n-grams have problems with representing patterns over more than a few words
3. With increasing order of the n-gram model, the number of possible parameters increases exponentially
4. There will be never enough of the training data to estimate parameters of high-order N-gram models

In order to model unseen sequences, N-gram models typically add smoothing which have been quite successful. For instance, Kneser-Ney smoothed 5-gram models have challenged few of the other parametric approaches including neural networks.

Gated Recurrent Units and Long Short-Term Memory

For our neural network architectures, we have experimented with Gated Recurrent Units and Long Short-Term Memory which are forms of Recurrent Neural Networks which can retain long term dependencies and do not suffer from vanishing gradient problems. This is achieved by internal mechanism called gates [9] which control the flow of information. These gates can learn which information in data is important and which can be thrown away. The architectures that we considered are highlighted in the subsequent sections.

word2Vec

For one of our neural networks, we train the model on the word embeddings for the input sequence of words and predict the word embedding of the next word. A word2vec model is used to construct both the input word embeddings as well as to derive the next word based on the predicted embedding. word2vec [10] is a popular word embedding technique which learns the distributed representation of words using neural nets. The learnt vectors are able to maintain linear semantic and syntactic relationships. There are two types of models called CBOW (cumulative bag of words) and Skip Gram. For our analysis, we fit a skip-gram model using hierarchical softmax to reduce the word count complexity.

3 Dataset

The Project Gutenberg corpus (http://www.gutenberg.org/) was considered for our analysis. Project Gutenberg is a library of over 60,000 free eBooks. The books in the project repository have been chronologically assigned a serial number which goes from 1 to ~62000. All files are stored as “UTF-8” encoded txt files. We have considered books from serial number 45,000 to 62,000 for our analysis. Once the books are downloaded, the following operations are performed to create the final corpus:

1. Books are filtered for English books
2. Book metadata and the Gutenberg license is removed from all the books’ texts
3. First 500 books’ texts are concatenated
4. The final corpus is cleaned by removing the occurrences of unwanted characters like ‘\n’ and multiple spaces between words
5. The sentences were padded with begin and end of sentence tokens for the N-gram modelling

After creating the final corpus, the text is tokenized using NLTK word_tokenize and sent_tokenize. The final corpus contains 186,928,914 characters and 39,120,715 words.

For the character level and word level neural nets, the training corpus was restricted to the first 1,800,000 units (characters for character level neural net and words for word level neural net) and the validation set was the next 200,000 units. The full corpus was taken for the N-Gram models.

4 Method

The following three models were fit on the data for language modelling:

N-Gram models

We trained four N-Gram models for N = 2, 3, 4 and 5 using NLTK lm module. This basic module does not perform any smoothing and hence outputs the probabilities for unseen sequences as zero. As a

result, we were unable to calculate the perplexity metric which is generally used to report performance of language models.

Character level Neural Net

Figure 1 depicts the model architecture used for character level neural net model. The input is a sequence of 128 characters and the model aims to predict the next character. The model consists of a Keras embedding layer which creates embeddings of dimension 128 for the input characters. This is followed by two or more recurrent neural network layers with varying hidden states. The final layer is a softmax dense layer over the entire character set which calculates the probability of the next character. There are 157 unique characters in our training set over which the model calculates the probability. Different model architectures are constructed by varying the number of hidden states of the RNN, number of RNN layers as well as type of RNN unit (LSTM or GRU). The loss that is minimized is categorical cross-entropy. All the models are trained for 5 epochs and the training time and validation cross entropy is recorded.

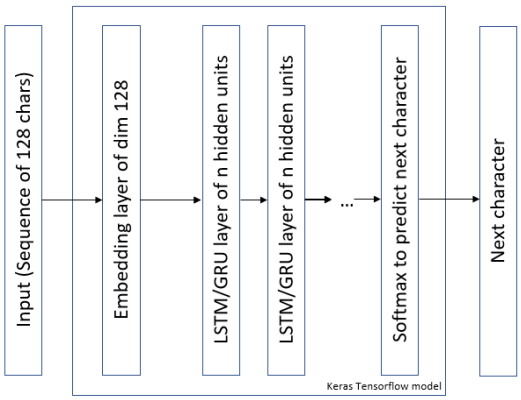

Figure 1

Word level Neural Net

The biggest challenge with constructing a word level neural net is that the final softmax layer needs to predict over the entire vocab which may contain millions of words and hence becomes computationally intensive. Although there are techniques to count this like Noise Contrastive Estimation (NCE) loss, self-normalizing partition functions, or hierarchical softmax, we decided to adopt a slightly different approach. Instead of modelling the probability of the next word, we modelled the word embeddings. Figure 2 shows the model architecture. First, a word2vec model is fit on the entire training corpus. A sequence of 128 words is passed through the word2vec model to get the embeddings which form the input for the neural net model. This is followed by two or more recurrent neural network layers with varying hidden states. The final layer is a tanh dense layer of dimension 300 which predicts the embedding of the next word. The loss which is optimized for is mean square error. Different model architectures are constructed by varying the number of hidden states of the RNN, number of RNN layers as well as type of RNN unit (LSTM or GRU). All the models are trained for 5 epochs and the training time and validation mse is recorded.

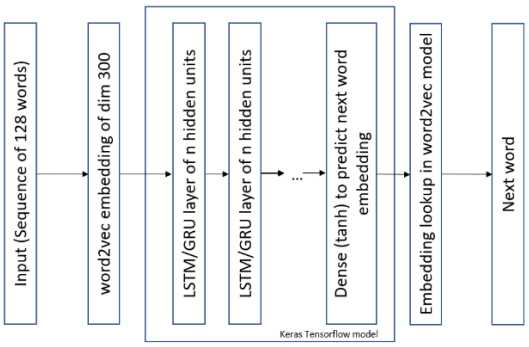

Figure 2

Training Procedure

Dropout of 0.3 and a l2 penalty of 0.0003 was used in all neural net models. All models were trained in batches of 250 for a total of 5 epochs on a GeForce GTX 1080 8gb dedicated GPU.

5 Results

N-Gram models

Table 1 shows the Train Time and the Model size for the N-Gram models. As it is seen, the models do not take substantial time to fit, however as the N count increases, the model parameters and hence the model size increases exponentially. As it is noted before, since our model does not incorporate any smoothing, we are unable to report any performance metrics like perplexity.

Model	N-Grams	Train Time	Model Size
Model 1	2	8min	110mb
Model 2	3	14min	513mb
Model 3	4	23min	1.40gb
Model 4	5	34min	2.81gb

Table 1

Character level Neural Net

Table 2 contains the details for the various character level neural net models trained and their performance metrics. As it is seen, the best

performing model after 5 epochs is the model with two LSTM layers having 1024 hidden units. One important point to note is that this model has the highest number of parameters and took the most amount of time to train.

Model	LSTM/GRU	Hidden Units	RNN Layers	# of params	Train Time (5 Epochs)	Val cross-entropy
Model 1	LSTM	256	2	980,125	3hrs15 min	1.61
Model 2	LSTM	1,024	2	13,296,541	8hrs 40min	1.56
Model 3	LSTM	256	5	2,556,061	7hrs 26min	1.68
Model 4	GRU	256	2	750,237	2hrs 20min	1.63
Model 5	GRU	1,024	2	10,017,693	6hrs16min	1.70
Model 6	GRU	256	5	1,932,189	7hrs 21min	2.19

Table 2

Word level Neural Net

Table 3 contains the details for the various word level neural net models trained and their performance metrics. As it is seen, the best performing model after 5 epochs is the model with two LSTM layers having 1024 hidden units. While this model also contains the maximum number of parameters, this model is not the most time-consuming model to train which is the model with GRU units.

Model	LSTM/GRU	Hidden Units	RNN Layers	# of params	Train Time (5 Epochs)	Val mse
Model 1	LSTM	256	2	1,172,780	4hrs 24min	0.0197906
Model 2	LSTM	1,024	2	14,127,404	8hrs 58min	0.0197046
Model 3	LSTM	256	5	2,748,716	10hrs 24min	0.0199531
Model 4	GRU	256	2	898,860	3hrs 50min	0.0198417
Model 5	GRU	1,024	2	10,672,428	6hrs 54min	0.0197907
Model 6	GRU	256	5	2,080,812	6hrs 6min	0.0199277

Table 3

6 Discussion

Literature review mentioned that for N-Gram models, as we increase the N count, the performance improves. This was not necessarily observed in our case. A 2-Gram model output looked like - *"All at once, I saw two figures: one a little man who was stumping along eastward at a good walk, and the other a girl of maybe eight or ten who was running as hard as she was able down a cross street. , (p. 212 . , him , I have law , hollowed out very characteristic was his keeping a road , and Aline and who would not , whose support the animus , broad stone from the broad enough to the sea-like arena to belong to mention another man himself with Jeremiah , " corporal , and most common faith , protected .] rears me tell me not so it , let it more importance usually calm of the lives , THESE two daughters , and more new , " , The right , for in the*" where the text in black is input and the text in red is the predicted output predicting for 100 words. The same input did not yield any output for models trained on longer sequences. The primary reason for this was that we did not implement any smoothing and the model could not assign probabilities to sequences it had not seen and hence predicted nothing.

The 2-layer LSTM network with 1,024 hidden units performed the best for both the character as well as the word level neural net models. However, a quick look at some of the results extracted from these models revealed that the models had not really converged and learnt the intricate structures and the long-term dependencies. For instance, an output of the character level neural net model looks like: *"All at once, I saw two figures: one a little man who was stumping along eastward at a good walk, and the other a girl of maybe eight or ten who was running as hard as she was able down a cross street. In the Duke of Bedford and Lord Holland was a man of the same time to the same time to the same time to the same time to the contempt of the House of Commons, and the contempt was the same time to the"*. Similarly, an output of the best word level neural net model is *"All at once, I saw two figures: one a little man who was stumping along eastward at a good walk, and the other a girl of maybe eight or ten who was running as hard as she was able down a cross street. Well, sir, the two ran into one another naturally enough at the corner; and then came the horrible part of the thing; for the man trampled calmly over the childs body and left her screaming on the ground. It sounds nothing to hear, but it was hellish to see. It wasn't like a man; it was like some damned Juggernaut. I gave a few halloa, took to my heels, collared my gentleman, and brought him back to where there was already quite a group about the screaming child. He was perfectly cool and made no resistance, but gave me one look, so ugly that it brought out the sweat on me like running. " : that s and and and and and and and and and and and and and and and and and and and and and and and and and and and and and and and and and and and and and and and and"*. Overall, for the same parameters, the character level neural net performs better than the word level neural net. As it is seen, the models have learnt the word structures, however have not yet learnt the sentences semantics and will have to be trained for longer epochs for better performances.

As a few next steps to improve the models, it is recommended that smoothing be introduced in the N-gram models and the neural network models be trained for longer epochs so that they are able to learn the intricacies of the language semantics and perform better.

References

[1] Mikolov, Tomas, Karafi´at, Martin, Burget, Lukas, Cernock`y, Jan, and Khudanpur, Sanjeev. Recurrent neural network based language model. In INTERSPEECH, volume 2, pp. 3, 2010.

[2] Arisoy, Ebru, Sainath, Tara N, Kingsbury, Brian, and Ramabhadran, Bhuvana. Deep neural network language models. In Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT, pp. 20–28. Association for Computational Linguistics, 2012.

[3] Rush, Alexander M, Chopra, Sumit, and Weston, Jason. A neural attention model for abstractive sentence summarization. arXiv preprint arXiv:1509.00685, 2015.

[4] Filippova, Katja, Alfonseca, Enrique, Colmenares, Carlos A, Kaiser, Lukasz, and Vinyals, Oriol. Sentence compression by deletion with lstms. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 360–368, 2015.

[5] Schwenk, Holger, Rousseau, Anthony, and Attik, Mohammed. Large, pruned or continuous space language models on a gpu for statistical machine translation. In Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT, pp. 11–19. Association for Computational Linguistics, 2012.

[6] Advances in neural information processing systems, pp. 3104–3112, 2014. Vaswani, Ashish, Zhao, Yinggong, Fossum, Victoria, and Chiang, David. Decoding with large-scale neural language models improves translation. Citeseer.

[7] Srivastava, Nitish, Mansimov, Elman, and Salakhutdinov, Ruslan. Unsupervised learning of video representations using lstms. arXiv preprint arXiv:1502.04681, 2015a.

[8] Mikolov, Tom´aˇs. Statistical language models based on neural networks. Presentation at Google, Mountain View, 2nd April, 2012.

[9] Gers, Felix A, Schmidhuber, J¨urgen, and Cummins, Fred. Learning to forget: Continual prediction with lstm. Neural computation, 12(10):2451–2471, 2000.

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In NIPS, pages 3111–3119.