EasyChair Preprint

№ 623

LSTM Neural Network for Textual Ngrams

Shaun D'Souza

November 10, 2018

LSTM Neural Network for Textual Ngrams

Shaun C. D'Souza[1]

Accenture
shaun.c.dsouza@accenture.com

Abstract

Cognitive neuroscience is the study of how the human brain functions on tasks like decision making, language, perception and reasoning. Deep learning is a class of machine learning problems that use neural networks. They are designed to model the responses of neurons in the human brain. Learning can be supervised or unsupervised. Ngram token models are used extensively in language prediction. Ngrams are probabilistic models that are used in predicting the next word or token. They are a statistical model of word sequences or tokens and are called Language Models or Lms. Ngrams are essential in creating language prediction models. We are exploring a broader sandbox ecosystems enabling for AI. Specifically, around Deep learning applications on unstructured content form on the web.

Keywords— Deep learning, Cognitive, LSTM, Neural network, Ngrams

1 Introduction

Ngram models work on the basis that we can predict the next token given the previous n-1 tokens. We use the following notation to compute the probability of a word sequence. In order to represent a random variable X taking on the value y we use P(X = "y") or the simplification P(y). Now, to compute the joint probability of a sequence of words $w_1...w_n$ we use $P(w_1, w_2, ..., w_n)$.

We compute the probability of an entire sequence of word by decomposing this probability using the chain rule of probability

$$P(w_1, w_2, ..., w_n) = P(w_1|w_2, ..., w_n) * P(w_2|w_3, ..., w_n) * P(w_3|w_4, ..., w_n) * P(w_n) \tag{1}$$

The chain rule shows the relation between computing the joint probability of a sequence of words given the conditional probability of the previous sequence of words. Using a Ngram model allows us to further simplify this equation as we estimate the probability of a word given its history by approximating the last N words. A Bigram model for example would use the conditional probability of a word given the word before it.

$$P(w_1|w_2, ..., w_n) \approx P(w_1|w_2) = \frac{C(w_1, w_2)}{C(w_2)} \tag{2}$$

A trigram model allows us to improve our predictability by using the preceding 2 word tokens.

$$P(w_1|w_2, ..., w_n) \approx P(w_1|w_2, w_3) * P(w_2|w_3) \tag{3}$$

The easiest way to estimate these probabilities is using the count value of the token sequences in the training data.

$$P(w_1|w_2, ..., w_n) = \frac{C(w_1, w_2, ..., w_n)}{C(w_2, ..., w_n)} \tag{4}$$

To obtain the count values for our tokens we use the ngram utility [12] to obtain a set of token sequence counts for all the data in the training set. For the purposes of our investigation we used the textual book data from the Gutenberg project and Brown data set. The Google Books Ngram Corpus [10] is available at http://books.google.com/ngrams.

Total 9624 unique ngrams in 890415 1-grams	
the	50869
N	32481
of	24406
to	23662
a	21639

Table 1: 1-grams

Total 273906 unique ngrams in 890414 2-grams	
N_N	7204
of_the	5293
in_the	4507
N_million	4493
to_N	2865

Table 2: 2-grams

We process the data to obtain the 5-gram tokens using the ngram utility which gives us a count value of all consequent tokens in the training data. We are using textual data for the purposes of our investigation on computing the ngram probabilities.

We use a variety of smoothing techniques to normalize the data and since a large number of token sequences are not in the training data. We use a ngram log probability (NGLP) to estimate the probability of our language model. This allows us to maintain a sum value of the log probability for the training data as it is difficult to compute accumulated product value on decimal values. We use these values to compute the cross entropy of the data.

$$Crossentropy = H = -(log2(P(w_1|w_2, w_3) + log2(P(w_2|w_3, w_4)) + ...log2(P(w_n - 2|w_n - 1, w_n)) \quad (5)$$

This allows us to compute the cross entropy of the data on the test set which gives us a measure of how well our language model is able to predict the tokens in the code. We calculate a perplexity value equivalent to 2^H.

2 Results

A low cross entropy means that we are able to accurately predict the next token. If the model predicts every token correctly with a probability of 1, then our cross entropy is 0. We use this data to study

Total 613476 unique ngrams in 890413 3-grams	
the_u_s	920
N_million_or	687
of_N_million	665
N_a_share	631
million_or_N	621

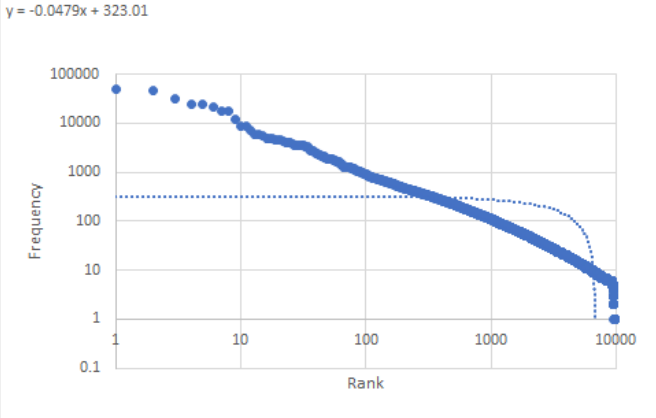
Table 3: 3-grams

3-grams	entropy	7.95
	perplexity	247

Table 4: Cross entropy

multiple types of identifiers in the code including variable and class field definitions, method names and function calls in the code.

Figure 1: Zipf distribution.



We measured the zipf distribution - Fig. 1 of the data in the text [3]. As per Zipf's law we see that the frequency of the tokens in the data set is inversely proportional to its rank in the number count of tokens. The training data contained 9624 unique tokens. The slope is -0.0479.

We plan to extend this code to deep learning applications [6, 7] on unstructured content form on the web along the lines of the Google Brain project [2] and TensorFlow [1]. This will allow us to build a knowledge base [14] using existing projects and reuse code as per the application.

TensorFlow is an open source library for deep learning developed by Google. It is a python library that is similar to numpy, scipy and uses data flow graphs and tensors for numerical computation. They support the development of neural networks using a set of libraries. A perceptron is a simple neural network designed to use a threshold activation function. It computes the activation of a neuron using the dot product of the input and weight vectors.

$$O(x) = sgn(\sum_{i=0}^{n} w_i * x_i) \tag{6}$$

$$\text{where } sgn(y) = \begin{cases} 1 & \text{if } y > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Fig. 2 shows a single layer perceptron. For a given dataset the perceptron is guaranteed to find a linear plane of separation described as the hyperplane decision surface in the n-dimension space. The perceptron training rule is used iteratively to update the network weights. The weight vectors are initialized randomly and updated using the rule

$$\begin{aligned} w_i &\leftarrow w_i + \Delta w_i \\ \Delta w_i &= \eta(t - o)x \end{aligned} \tag{7}$$

Additionally, multiple layers can be used in a multi-layer perceptron. This is effective on uni-dimensional data and finds application in a number of natural language processing tasks [4, 11] including

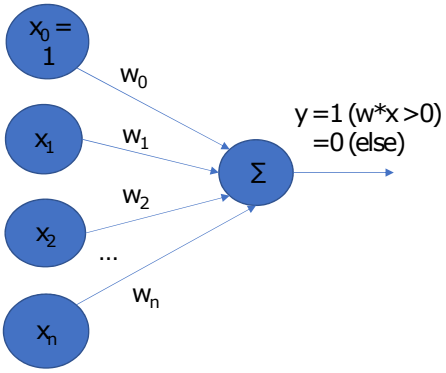


Figure 2: Single layer perceptron.

Sigmoid	Softmax	Hyperbolic tangent
$\sigma(y) = \frac{1}{1+e^{-y}}$	$f(x) = \frac{e^x}{\sum_{j=1}^{N} e^{x_j}}$	$tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$

Table 5: Activation functions

part of speech (POS) tagging. The OpenNLP library uses a multi-layer perceptron in its trained model. These have been effectively used in applications in information extraction as shown in [5]. [8] demonstrated the use of a chunker model in detection of semantic triples.

A neural network uses a continuous activation function in each of the layers. Some of the activation functions are in Table 5.

Figure 3: Neural Network.

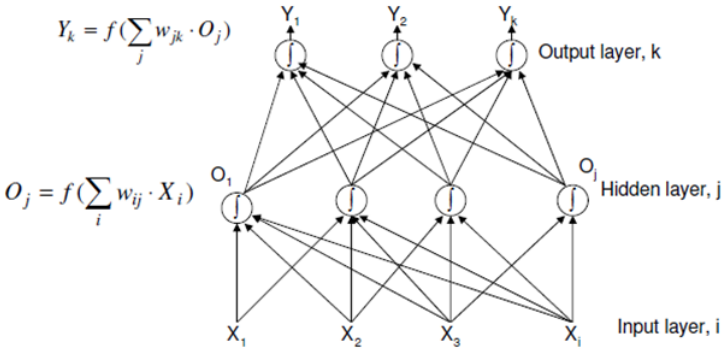


Fig. 3 shows an artificial neural network with Sigmoid activations. The weights in the hidden layer are used in determination of word vectors used in the Continuous Bag-of-words (CBOW) and Skip-gram models [13]. These are used in predicting semantic similarity. We use a TensorFlow Keras LSTM layer - Fig. 4 to output word sentences in the PTB corpus. Below is a generated text sequence corresponding to an input seed value. This can be used in applications like the Google Smart compose for autocompletion of input text given a suitable corpus of training data.

```
----- Generating with seed: "n plant near <unk> ill. was completed in n <eos> in a
```

```
    disputed n ruling the commerce commission said commonwealth edison could raise its
    electricity rates by $ n million to pay for the plant <eos> but state courts upheld
    "

n plant near <unk> ill. was completed in n <eos> in a disputed n ruling the commerce
    commission said commonwealth edison could raise its electricity rates by $ n
    million to pay for the plant <eos> but state courts upheld a challenge of last year
     's that list of the low 's action action of the low are to debt higher prices the
    in financing <eos> the u.s. until that the latest will leave <eos> the and and
    japan is expected to slip <eos> the funds have money funds have received these rich
     since that the decline plot increase <unk> with the low and japan automobile said
    that the u.s. is that that u.s. is is already for a request <eos> the decline and
    japan of their program managers the highest has authority dropped n n in september
    <eos> first n n for export and $ n million debt and year plant <eos>
```

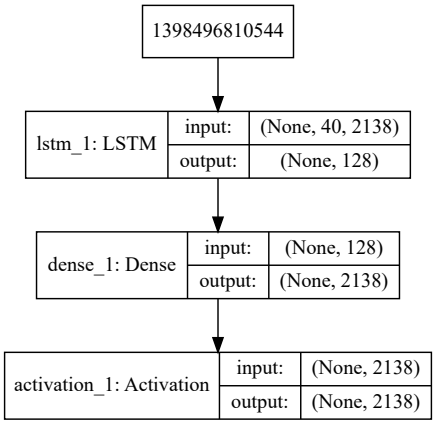


Figure 4: Keras LSTM model.

3 Conclusion

We explore the use of the TensorFlow library in creating a recurrent neural network (RNN). We train a LSTM neural network on textual data from the Penn Tree bank corpus. We see that the LSTM is able to accurately predict the word ngrams using a seed sentence. We plan to extend the work to use POS and chunker sequences [9] in phrase construction. We continue to explore the intersection of Technology and Business in the context of AI, Globalization, CSR and the Last mile with an emphasis on Deep learning applications in the broader web.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker,

Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[3] Miltiadis Allamanis and Charles Sutton. Mining source code repositories at massive scale using language modeling. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, pages 207–216. IEEE Press, 2013.

[4] Xavier Carreras and Lluís Màrquez. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the ninth conference on computational natural language learning*, pages 152–164. Association for Computational Linguistics, 2005.

[5] Shaun D'Souza. Parser extraction of triples in unstructured text. *IAES International Journal of Artificial Intelligence (IJ-AI)*, 5(4):143–148, 2017.

[6] Shaun C D'Souza. Cognitive architecture for a connected world. *arXiv preprint arXiv:1810.03955*, 2018.

[7] Shaun C D'Souza. Evolving system bottlenecks in the as a service cloud. *arXiv preprint arXiv:1809.07794*, 2018.

[8] Shaun Cyprian D'souza. System and method for extracting information from unstructured text, June 19 2018. US Patent App. 15/474,194.

[9] Matthew Honnibal and Mark Johnson. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[10] Yuri Lin, Jean-Baptiste Michel, Erez Lieberman Aiden, Jon Orwant, Will Brockman, and Slav Petrov. Syntactic annotations for the google books ngram corpus. In *Proceedings of the ACL 2012 system demonstrations*, pages 169–174. Association for Computational Linguistics, 2012.

[11] Donald Metzler and Oren Kurland. Experimental methods for information retrieval. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 1185–1186. ACM, 2012.

[12] Jean-Baptiste Michel, Yuan Kui Shen, Aviva P Aiden, Adrian Veres, Matthew K Gray, Joseph P Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, et al. Quantitative analysis of culture using millions of digitized books. *science*, page 1199644, 2010.

[13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[14] Rajeev Rastogi. Building knowledge bases from the web. In *Proceedings of the 18th International Conference on Management of Data*, pages 5–5. Computer Society of India, 2012.