Language modeling to predict the next best word using recurrent neural networks LSTM

Bhargavkumar Patel
dept. of Computer Science
Lakehead University
1106925
bpatel22@lakeheadu.ca

Vashishtha Master
dept. of Computer Science
Lakehead University
1100901
vmaster@lakeheadu.ca

Karan Atulkumar Shah
dept. of Computer Science
Lakehead University
1100690
kshah8@lakeheadu.ca

Pavankumar Patel
dept. of Computer Science
Lakehead University
1107159
ppatel59@lakeheadu.ca

***Abstract*— *Language Modeling (LM)* is one of the most important parts of modern Natural Language Processing (NLP). There are many sorts of applications for Language Modeling, like: Machine Translation, Spell Correction Speech Recognition, Summarization, Question Answering, Sentiment analysis etc. Each of those tasks require use of *language model.* Language model is required to represent the text to a form understandable from the machine point of view. Here, we have used the "ALICE'S ADVENTURES IN WONDERLAND Lewis Carroll, THE MILLENNIUM FULCRUM EDITION 3.0 "as the test based corpus to implement our project idea. The idea behind the implementation is to predict the next words from the text based corpus and to implement a model based on RNN LSTM to train and test the data.**

Keywords*— *RNN, LSTM, Language Modelling (LM), corpus, Natural Language Processing, Token.

I. INTRODUCTION

The goal of language modelling is to estimate the probability distribution of various linguistic units, e.g., words, sentences (Rosenfeld, 2000). Among the earliest techniques were count-based n-gram language models which intend to assign the probability distribution of a given word observed after a fixed number of previous words. Later Bengio et al. (2003) proposed feed-forward neural language model, which achieved substantial improvements in perplexity over count-based language models. Bengio et al. showed that this neural language model could simultaneously learn the conditional probability of the latest word in a sequence as well as a vector representation for each word in a predefined vocabulary.Recently recurrent neural networks have become one of the most widely used models in language modelling. Long short-term memory unit (LSTM, Hochreiter and Schmidhuber, 1997) is one of the most common recurrent activation function. Architecturally, the memory state and output state are explicitly separated by activation gates such that the vanishing gradient and exploding gradient problems is avoided. When modelling a corpus, these language models assume the mutual independence among sentences, and the task is often reduced to assigning a probability to a single sentence. In this work, we propose a method to incorporate corpus-level discourse dependency into neural language model. We call this larger-context language model. It models the influence of context by defining a conditional probability in the form of P(wn|w1:n−1, S), where w1, ..., wn are words from the same sentence, and S represents the context which consists a number of previous sentences of arbitrary length.
We proposed a late fusion approach, which is a modification to the LSTM such that it better incorporates the discourse context from preceding sentences. In the experiments, we evaluated the proposed approach against early fusion approach with various numbers of context sentences, and demonstrated the late fusion is superior to the early fusion approach.

II. STATISTICAL LANGUAGE MODELLING WITH RECURRENT NEURAL NETWORKS

Given a document D = (S1, S2. . . SL) which consists of L sentences, statistical language modelling aims at computing its probability P (D). It is often assumed that each sentence in the whole document is mutually independent from each other:

$$P(D) \approx \prod_{l=1}^{L} P(S_l). \tag{1}$$

We call this probability (before approximation) a corpus-level probability. Under this assumption of mutual independence among sentences, the task of language modelling is often reduced to assigning a probability to a single sentence P(Sl). A sentence Sl = (w1, w2, . . . , wTl) is a variable-length sequence of words or tokens.

$$P(S) = \prod_{t=1}^{T_l} p(w_t|w_{<t}), \tag{2}$$

By assuming that a word at any location in a sentence is largely predictable by preceding words, we can rewrite the sentence probability into P(S) = Y Tl t=1 p(wt |w<t), (2) where w denotes all the preceding words. We call this a sentence-level probability.
This rewritten probability expression can be either directly modelled by a recurrent neural network (Mikolov et al., 2010) or further approximated as a product of n-gram conditional probabilities such that

$$P(S) \approx \prod_{t=1}^{T_l} p(w_t | w_{t-(n-1)}^{t-1}), \tag{3}$$

where w t−1 t−(n−1) = (wt−(n−1), . . . , wt−1). The latter is called n-gram language modelling. A recurrent language model is composed of two functions–transition and output functions. The transition function reads one word wt and updates its hidden state such that

$$\mathbf{h}_t = \phi\left(w_t, \mathbf{h}_{t-1}\right), \tag{4}$$

where ht is an all-zero vector. φ is a recurrent activation function. For more details on widelyused recurrent activation units, we refer the reader to (Jozefowicz et al., 2015; Greff et al., 2015). At each timestep, the output function computes the probability over all possible next words in the vocabulary V .This is done by

$$p(w_{t+1} = w' | w_1^t) \propto \exp\left(g_{w'}(\mathbf{h}_t)\right). \tag{5}$$

g is commonly an affine transformation:

where Wo ∈ R |V |×d and bo ∈ R |V | . The whole model is trained by maximizing the log-likelihood of a training corpus often using stochastic gradient descent with backpropagation through time (see, e.g., Rumelhart et al., 1988). This conventional approach to statistical language modelling often treats every sentence in a document to be independent from each other This is often due to the fact that downstream tasks, such as speech recognition and machine translation, are done sentence-wise. In this paper, we test the accuracy of how well the next words can be predicted by using some of the traditional concepts of N-Grams and probability.

III. N-GRAMS

N-grams (6) of texts are extensively used in text mining and natural language processing tasks. They are basically a set of co-occurring words within a given window and when computing the n-grams you typically move one word forward (although you can move X words forward in more advanced scenarios).The use of n-grams is for developing features for supervised Machine Learning models such as SVMs, LSTM models, Naive Bayes, etc. The idea is to use tokens such as bigrams in the feature space instead of just unigrams.

IV. LONG SHORT TERM MEMORY

Here let us briefly describe a long short-term memory unit which is widely used as a recurrent activation function φ (see Eq. (4)) for language modelling (see, e.g., Graves, 2013). A layer of long short-term memory (LSTM) unit consists of three gates and a single memory cell. This adaptive leaky integration of the memory cell allows the LSTM to easily capture long-term dependencies in the input sequence.

The output, or the activation of this LSTM layer, is then computed by ht = ot tanh(ct).

V. LITERATURE REVIEW

In the field of natural language processing we came up with various techniques and methodologies till this date. In 2001, the first advancement was observed in the field of natural language processing. Language modelling was the term which was used at that period. Basically, language modelling is the method of predicting next word. It is probably the simplest task in language processing. You can see the application of these recently in Spelling Correction, Intelligent Keyboards, Email response suggestion, etc. The statistical models are the distribution of probabilities over the sequence of words. Statistical language models play an important role in the large vocabulary continuous speech recognition (LVCSR) systems. However, the Feed forward neural networks (FFNN) in the recent year showed how it can overcome the biggest disadvantage of n-gram models. The input vector of this model is represented by the n words above. Such vectors are now referred to as word embedding. These word embedding are concatenated and fed into a hidden layer with a Softmax layer output.

Recently, the above viewed FFNN were replaced by Recurrent Neural Network (RNN) and the Long Short Term Memory (LSTM) [7,8]. The previous FFNN model used to take all the previous words in to consideration to predict the next word, while the recent RNN models predicts the next based on the few previous words. LSTM model structure allows to discover both the long and short pattern in the data. This model also solves the problem of vanishing gradient generated by RNN model. Recurrent neural networks are the obvious choice when you have to deal with the dynamic input sequence ubiquitous in NLP.

The vanishing gradient in RNN proved to be a troublesome case. It was observed that each time the gradient of the error function of the neural network is propagated back through a unit of the neural network it is scaled by some factor. In almost all technically practicable circumstances this dimension is either greater than one or less than one. Consequently, in a recurrent neural network, the gradient increases or decreases exponentially over time. The gradient thus either dominates the next weight improvement stage, or is effectively lost.

Thus it led the authors to redesign the LSTM network unit. A layer of long short-term memory (LSTM) unit consists of three gates and a single memory cell. Three gates–input, output and forget.

$$\begin{aligned}
\mathbf{i}_t &= \sigma\left(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i\right) \\
\mathbf{o}_t &= \sigma\left(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o\right) \\
\mathbf{f}_t &= \sigma\left(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f\right)
\end{aligned}$$

where σ is a sigmoid function. x_t is the input at the t-th timestep.

Figure 1 showed that each LSTM cell has gates to decide when the input is large enough to remember, when it should keep remembering or forgetting about the value, and when the value should be output.

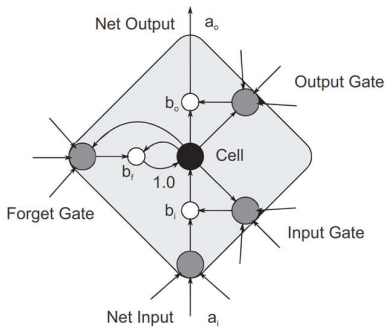


Figure 1: LSTM memory gates with cell[9]

VI. PROPOSED MODEL

Predicting the next word based on some typed words has many real-word applications. An example would be to suggest the word while typing it into the Google search bar. This type of feature does improve user satisfaction in using search engines. Technically, this can be called **N-grams** (if two consecutive words are extracted, it will be called **bi-grams**). Though there are so many ways to model this, here we have chosen deep RNNs to predict the next best word based on *N-1* pre-words.

Alice in Wonderland data has been used for this purpose and the same data can be downloaded from http://www.umich.edu/~umfandsf/other/ebooks/alice30.txt. In the initial data preparation stage, we have extracted N-grams from continuous text file data, which looks like this

```
# File Reading
with open('/content/alice_in_wonderland.txt','r') as content_file:
  content = content_file.read()
content2 = " ".join("".join([" "if ch in string.punctuation else ch for ch in content]).split())
tokens = nltk.word_tokenize(content2)
tokens = [word.lower() for word in tokens if len(word)>=2]
```

N-grams are selected with the following N value. In the following code, we have chosen N as 3, which means each piece of data has three words consecutively. Among them, two pre-words (bi-grams) used to predict the next word in each observation. Readers are encouraged to change the value of N and see how the model predicts the words

```
# N value = 3 for N grams, in which N-1 are used to predict last Nth word
N = 3
quads = list(nltk.ngrams(tokens,N))
newl_app = []
for ln in quads:
  newl = " ".join(ln)
  newl_app.append(newl)
```

After extracting basic data observations, we need to perform the following operations:

1. **Preprocessing**: In the preprocessing step, words are converted to vectorized form, which is needed for working with the model.
2. **Model development and validation**: Create a convergent-divergent model to map the input to the output, followed by training and validation data.
3. **Prediction of next best word**: Utilize the trained model to predict the next best word on test data.

Vectorization of the given words (*X* and *Y* words) to vector space using CountVectorizer from scikit-learn:

```
# Vectorizing the words

from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
x_trigm = []
y_trigm = []
for l in newl_app:
  x_str = " ".join(l.split()[0:N-1])
  y_str = l.split()[N-1]
  x_trigm.append(x_str)
  y_trigm.append(y_str)
x_trigm_check = vectorizer.fit_transform(x_trigm).todense()
y_trigm_check = vectorizer.fit_transform(y_trigm).todense()

# Dictionaries from word to integer and integer to word

dictnry = vectorizer.vocabulary_
rev_dictnry = {v:k for k,v in dictnry.items()}
X = np.array(x_trigm_check)
Y = np.array(y_trigm_check)
Xtrain, Xtest, Ytrain, Ytest,xtrain_tg,xtest_tg = train_test_split(X, Y,x_trigm, test_size=0.3,random_state=42)
print("X Train shape",Xtrain.shape, "Y Train shape" , Ytrain.shape)
print("X Test shape",Xtest.shape, "Y Test shape" , Ytest.shape)
```

After converting the data into vectorized form, we can see that the column value remains the same, which is the vocabulary length (2562 of all possible words):

```
print("X Train shape",Xtrain.shape, "Y Train shape" , Ytrain.shape)
print("X Test shape",Xtest.shape, "Y Test shape" , Ytest.shape)
```

```
X Train shape (17944, 2562) Y Train shape (17944, 2562)
X Test shape (7691, 2562) Y Test shape (7691, 2562)
```

Figure 2: Training and Testing Shape

The following code is the heart of the model, consisting of convergent-divergent architecture that reduces and expands the shape of the neural network.

This screenshot depicts the complete architecture of the model, consisting of a convergent-divergent structure. Different layers used like the dense layer ,dropout layer are used for the extracting the features of the data and to get better results.

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input (InputLayer)           (None, 2562)              0
_________________________________________________________________
first (Dense)                (None, 1000)              2563000
_________________________________________________________________
firstdout (Dropout)          (None, 1000)              0
_________________________________________________________________
second (Dense)               (None, 800)               800800
_________________________________________________________________
third (Dense)                (None, 1000)              801000
_________________________________________________________________
thirdout (Dropout)           (None, 1000)              0
_________________________________________________________________
fourth (Dense)               (None, 2562)              2564562
=================================================================
Total params: 6,729,362
Trainable params: 6,729,362
Non-trainable params: 0
_________________________________________________________________
```

Figure 2: RNN LSTM Model

VI.I Experimental Analysis

The model is trained on data with 100 epochs. Even after a significant improvement in the train accuracy (from 6.85% to 63.77%), there is little improvement in the validation accuracy (6.07% to 11.23%).

Appendix:

```
# Sample check on Test data
print ("Prior bigram words", "|Actual", "|Predicted","\n")
for i in range(50):
  print (i,xtest_tg[i],"|",rev_dictnry[np.argmax(Ytest[i])], "|",
rev_dictnry[np.argmax(Y_pred[i])])
```

VI.II Ouput

```
Prior bigram words |Actual |Predicted

0 and they | re | repeated
1 never learnt | it | seen
2 matters it | how | good
3 as she | could | could
4 the duchess | what | and
5 pair of | boots | white
6 cattle in | the | the
7 it was | looking | the
8 what else | have | are
9 voice of | the | the
10 to make | out | out
```

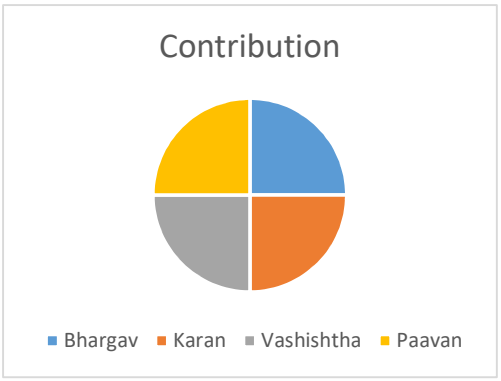
Figure 3: Implemented Output predicting the words

VII. Contribution

During the semester we have worked on two different applications of Natural Language Processing namely : Named Entity Representation and Language Modelling to predict the next best word from the corpus using RNN (LSTM) model and probabilistic N-Gram approach. During the first half of the semester we had implemented the Named Entity Representation and had gone through many research papers in order to get a brief knowledge on the methods available to perform NER(Named Entity Representation) and to find new ways to improvise the technique to predict different parts of speech from the given sentence. During the other half we have decided to work on some other application and so we selected Language Modelling as a part of our final project. N-Grams and probabilistic Language modelling were part of our curriculum and we were inclined to the fact that if given some text data we can predict the next words from the data and so we designed a model that uses RNN and LSTM approach to predict the next best words as per the training of the model. So to perform this we had a team of four members and we had divided the tasks amongst us in order to balance the work equally and in disciplined manner. The task were divided in four steps for designing the model:

1. Selection of dataset and designing of the overview and the flow of the program.
2. Designing the RNN model
3. Logic behind predicting the next word using the trained model
4. Formatting the code with improvements

Apart from this during the report we had equally divided the task of collection of informational content by reading different research paper and gathering the content. Everyone from our group has really worked hard and with discipline to complete the project as a team. If we look at the contribution with the numbers all four members would have worked 25 percent each of the total to be precise.



VIII. Acknowledgement

I would like to acknowledge that the efforts which were made by Professor Dr. Thangarajan Akilan was remarkable not only he introduced the topic in detail it has created huge interest for this subject in my mind. Also the work done by Andrew and Purandeep who were always available for help made by progress of learning the concept really easy.

IX. Conclusion

This project implementation aims to review recent studies and methods to solve problems with language modelling (LM) and to help new researchers building a comprehensive understanding of this field by summarizing the various methods and approaches into a single paper. By the end of the implementation we are able to predict the next words from the text based corpus and we implemented a model based on RNN LSTM to train and test the data. At last we were able to gain significant knowledge about language modelling (LM) and LSTM.

X. REFERENCE

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 .J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] Yoshua Bengio, Rejean Ducharme, Pascal Vin- ´ cent, and Christian Janvin. 2003. A neural probabilistic language model. The Journal of Machine Learning Research 3:1137–1155.K. Elissa, "Title of paper if known," unpublished.

[3] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. Neural Networks, IEEE Transactions on 5(2):157–166.Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[4] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. the Journal of machine Learning research 3:993–1022.

[5] Kyunghyun Cho, Bart Van Merrienboer, Caglar ¨ Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 .

[6] https://kavita-ganesan.com/what-are-n-grams/#.XoD8HYhKhPY

[7] Sepp Hochreiter and Jurgen Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp.1735–1780, 1997.

[8] Tomas Mikolov, Martin Karafiat, Luk ´ as Burget, Jan Cernocky, and Sanjeev Khudanpur, "Recurrent neural network based language model," in Proceedings of the 11th Annual Conference of the

[9] Soutner D., Müller L. (2013) Application of LSTM Neural Network In Language Modelling. In: Habernal I., Matoušek V. (eds) Text, Speech, and Dialogue.TSD 2013. Lecture Notes in Computer Sci, vol 8082. Springer, Berlin, Heidelberg