Journal of Industrial Information Technology and Application Vol. 2 No. 4

ISSN(Online) : 2586-0852

Building a Language Model for the Cebuano Language

John Vincent N. Pakson [1)], Robert R. Roxas [2,*)]

[1, 2)] Dept. of Computer Science, College of Science, University of the Philippines Cebu, Cebu City, Philippines

Abstract. This paper presents the result of building a language model for the Cebuano Language, which is greatly used in Central Visayas, Philippines. This language model is made possible using the Recurrent Neural Network (RNN) algorithm and a dataset of Cebuano articles translated from English articles by human translators and from online news articles in Cebuano. In this paper, we will also discuss the steps that we will be doing to determine the most suitable Language Model for Cebuano. The produced model expectedly generated a functional local language model that can be used in different aspects of today's technologies such as spelling correction, word completion, automatic speech recognition, and machine translation.

Keywords; language model; Cebuano Language; recurrent neural network

1. Introduction

Language Models are widely used in different technologies nowadays such as machine translation, speech recognition, spelling correction, and even sentence generation. Although language models are widely used, these models are largely based on high-density languages such as English or Chinese. This can be seen in the case of Google searches, Office applications, and most especially on many auto-suggest features on keyboard applications. The Cebuano Language is considered as a low-density language [1]. Because Cebuano language belongs to the low-density language, the

* Corresponding author: robert.roxas@up.edu.ph
Received: 2017.10.20; Accepted: 2018.3.15; Published: 2018.12.30

creation of its language model is a challenge. It means that there are lesser acquirable and usable resources for the language [2]. Although there have been studies on multilingual language models that involve the Cebuano language [3], there is still no considerable published researches on the aspect of creating a specific language model for the Cebuano Language. So, this paper presents our Cebuano Language model using the Recurrent Neural Network approach.

2. Related Works

The application of machine learning in natural language processing became widely used as it showed better performance than *N*-gram models. Proving this was the research by Bengio et al. in 2003, where they aimed to resolve the curse of dimensionality by creating a neural probabilistic language model [4]. Their study compared the performance of *N*-gram model against their own model. With the use of the Brown corpus, they trained both *N*-gram and neural network model to produce comparable results. As a result, the neural probabilistic model [4] showed better result with 10 to 20 percent perplexity improvement than that of a smoothed trigram model. This result may have been caused by the model's use of generalization through distributed representation of the data [4]. With these different models in mind, it would be unreasonable not to try to use the neural network approach in creating our own language model for Cebuano.

In creating a language model, the aim is to be able to predict the next word of a given context [5], which has been solved by significant researches on Parse Trees and *N*-grams. But then we are faced with the significant problem that the variety of grammar, semantics, and sentence constructions in different languages would increase the complexity of these models. This encourages us to work with better solutions in creating language models that could provide us better performance and less complexity when it comes to low-density languages.

Even with the lack of studies for language models specifically created for the Cebuano Language, language models for low-density languages have gained their own popularity in the field. Gandhe et al., however, had shown us the possibility of using neural network language model on low-density languages [6]. In their study, they showed the comparison between the use of the basic feed-forward neural network and recurrent neural network with the amount of data they had for the low-density languages. The amount of data provided much significance in their research as the feed-forward neural network showed better performance than that of recurrent neural network when the training data was less and in turn, the recurrent neural network performed better than the feed-forward neural network when the training data increased.

On a different note, there is still a wide range of variety when it comes to the use of neural networks. We could choose from a neural probabilistic neural language model [4] or a deep neural network language model [7]. Moreover, a variation of deep neural network introduced by Mikolov et al. as a solution for language modeling gives us a more specific path to take [5].

A more ideal type of neural network to be used in the case of creating a language is the state-of-the-art, recurrent neural network language model [5]. Recurrent neural network language models, as shown in Mikolov's study greatly outperforms the Word Error Rate (WER) and perplexity of its predecessor's state-of-the-art language models, the *N*-grams.

3. Methods

Building a language model for a certain language, especially for the low-density languages, involves several steps. So in building a language model for Cebuano, we performed several tasks as enumerated below:

A. Data Acquisition

With the knowledge that Cebuano is a low-density language, the initial step that we took was to determine probable resources for Cebuano articles to be used for the language model. Within the duration of this research, we were able to determine sources of Cebuano text data that would best help us build the language model.

The initial source of data that we had were 145 translated articles. These articles were English articles that were translated into Cebuano by human translators. This set of data provided us with an initial intuition on the data that we would be working with. This data set enabled us to work on certain algorithms that would parse data into the structure that we needed for modeling. This dataset is small but concise providing us with most of the possible sentence structure that we would encounter when creating the language model. The dataset gave us an opportunity to test the algorithms for pre-processing our data in building the language model, thus enabling us to revise these algorithms faster.

Wikipedia provides huge data dumps of the Cebuano translation of all its articles. After downloading the dump, it should then be subjected to post-processing. We chose to use an external python library called WikiExtractor [8]. With the use of this python library, we were able to acquire about 6 million articles from Wikipedia data dump. The number of articles acquired from Wikipedia was a lot of data to work with. Wikipedia provided us with too much data thus, about 10% of these articles will only be chosen as part of our dataset.

Even with the huge amount of Wikipedia data, we faced a problem that would limit the construction of our sentences, since Wikipedia's data contained articles that had sentences with repetitive patterns and were non-conversational. Fortunately, there were tons of online news databases that contained huge Cebuano news data such as Super Balita and Philstar's Banat News. We gathered the data using a python library called Scrapy that extracts the data from some parts of webpages.

B. Data Parsing

The initial data acquired, which was formatted as shown in Figure 1, is subjected to preprocessing, where they were converted to their raw string data. After parsing all the articles, all of them are now to be stored in a list, where we will then iterate through to tokenize the articles into sentences. To enable us to use the tokenized sentences in building our language model, we made sure that commas, quotations, and other punctuation marks were considered as words because we would also want the model to be able to predict when one of the symbols mentioned above should be used. It is important that we parse the sentences as advised by Britz [9]. Even with an existing code created by Britz', it was important that modification of the code was done with the variety of data that we had and with the idea that we were working with a different language and sentence formats.

```
<doc id = "6205298" url= "https://ceb.wikipedia.org/
wiki?curid="6205298" title = "Veihombergjae">

*article here*

</doc>

{"article": "*article here*",
"title": "Milagro"}
```

Fig 1. An unprocessed Wikipedia article and a JSON object of a new article respectively

C. Data Preprocessing

We then created our vocabulary ranging from 4,000 to 5,000 words because having a huge vocabulary would mean that our model would train slower [10]. We then added special tokens to our sentences such as our SENTENCE_START, SENTENCE_END, and UKNOWN_TOKEN. This UNKNOWN_TOKEN represents the unknown words found in our sentences that do not belong to our chosen vocabulary.

We also be made dictionaries of index to word and word to index relationships that we will use for prediction and in building the language model. These dictionaries were

significant for us since we tried to generate sentences. At the same time, we would be initializing our training data for our model.

D. Recurrent Neural Network with Back propagation through Time

With the popularity of Recurrent Neural Network Language Models (RNNLM), it would be good for us to use this in building our language model. This algorithm was also used by [6] in creating a language model on low-density languages. In checking the performance of the said language model, we used cross-entropy loss as was used by [11] in obtaining the error vector of their output layer. Moreover, to check the performance of the model, Sentence Generation was also applied. Figure 2 shows a typical recurrent neural network with backpropagation through time.

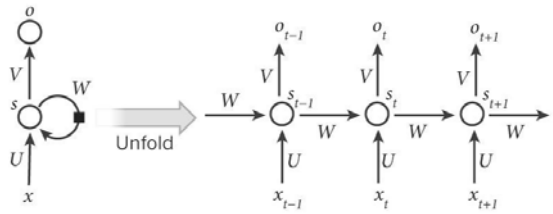

Fig 2. The unfolding of a recurrent neural network [10]

Although RNNLM's are popular, it still faces the problem of long-term dependencies, where it becomes harder for the model to predict and understand the context as the sequence become longer. Backpropagation Through Time (BPTT) was used by Mikolov et al. as a solution to this problem amongst RNN Language Models [11].

E. Cross-entropy Loss and Perplexity

The cross-entropy loss was used to check the performance of our model as we trained it as was used by Mikolov et al. in obtaining the error vector in their output layer [11].

$$L(y,o) = -\frac{1}{N}\sum_{n\in N} y_n \log o_n \qquad (1)$$

The formula for our cross-entropy loss is shown in (1), in calculating the error after every epoch of training the RNN [9]. Cross-entropy loss is the result of our cross-entropy function, which is mostly used as a solution to learning slowdown to most cost functions such as the quadratic cost function. The cross-entropy loss shows us the difference between predicted probabilities and ground-truth probabilities [12], which is more

important since ground-truth probabilities is what we want our models to predict. This means that the lesser our cross-entropy loss is, the closer we are to the actual or ground-truth probability, which is the correct prediction.

We will also calculate our perplexity based on our cross-entropy loss [13] as defined by Mohammadi et al., which is two to the power of our cross-entropy (J) as shown in (2).

$$Perplexity = 2^{J} \tag{2}$$

F. Sentence Generation.

Also, as another way to check the performance our trained model, we would touch another aspect of NLP which is text generation. We will be generating texts through the series of predictions of our language model. According to Bowman et al., RNNs produces the state-of-the-art results for sentence generation [14].

4. Results and Discussion

With the number of articles, we selected a vocabulary of 5,000 words, which had the most number of occurrences. The 5,000 words included the special tokens mentioned in our data preprocessing. The language model is then based on this vocabulary. To evaluate the performance of recurrent neural networks in building the language model for the Cebuano language, we built three different models for our network. We were then able to use 1,000, 10,000 and 50,000 sentences as training data initially to be able to create intuitions on the performance of the model. To train these models, it took about 2 days at most.

TABLE I. RESULTS OF MODEL TRAINING

Models	Cross-Entropy Loss
RNN with 1,000 sentences	3.557
RNN with 10,000 sentences	3.567
RNN with 50,000 sentences	3.623

The cross-entropy loss was calculated every epoch, this providing us with an insight if ever the model would be able to train well. As more epochs were added, the cross-entropy loss became smaller, thus telling us that there might be an increase in the performance of the model. The result of the training is shown in Table I. It seems that the results looked good as an initial result for our model. The results seem to have not much difference among the cross-entropy losses of the three models. As shown in Figure

3, during the first few epochs, the cross-entropy loss was significantly high. This means that the model's prediction probability is far from the actual sequence of words. But as more epochs have been done, the cross-entropy loss become very small and it prevails to be very low until the end. This means that the model has better predictions.

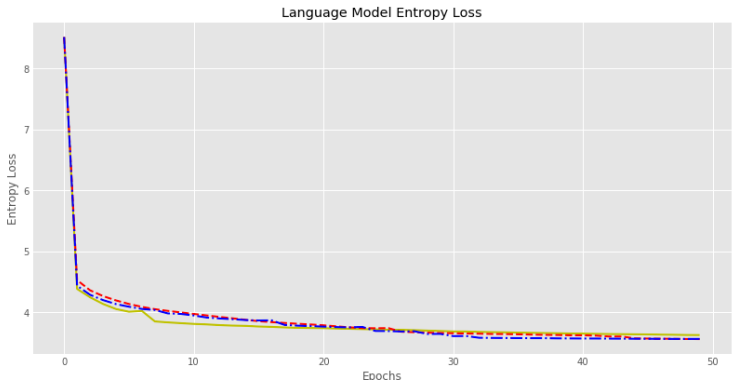

Fig 3. Cross-entropy loss of the three models throughout its training

Initially, we could say that the model has performed well in the aspect of its cross-entropy loss, but it seems that we have somehow ran the risk of overfitting due to the parameters we used in creating the language model. We could also observe that most of the sentences generated are in the context of the entertainment category of news articles. Also, the sentences generated were not completely correct but somehow the sequencing of words shows a relatively good result. Even with good results shown by the cross-entropy loss, the generated sentences built from the RNNLM were not yet very good. The naturalness of the sentences generated was not achieved but the arrangement of the words generated seemed like a good start. To further improve the performance of our models, we should use a larger set of training data and the application of more complex algorithms such as LSTM/GRU [15].

Sa maong episode naghisgot mao ang host.

Gibutyag sa awit ang pelikulang My Chinese Love.
Kahinumduman nga rason tali karon ni Presidente Duterte laing nasud.

Suma pa nga minyo si Cesar.

Fig 4. Sample generated sentences

As we can see in the results shown in Table I, there seems to be not much of a difference of the cross-entropy loss of the three models, which tells us that the difference of the number of examples seems to have less effect on the performance. Also, it would be important to remember that the sentences that we trained might have come from the

same category of news articles resulting to its current performance. To show this, we generated different sentences so as to further evaluate the model that we have created which is shown in Figure 4.

Even with great performance with regards to its cross-entropy loss, the generation of sentences from the built RNN Language Model performed badly. The naturalness of the sentences generated was not yet achieved but are close to normal sentences. These results might be due to the architecture used for the neural networks where as the number of layers and the number of nodes used per layer was given no variation. Moreover, the variety of data that was used for the models was also low thus creating sentences from the same news category of our training data.

5. Conclusion and Future Works

Building a Language Model for the Cebuano language has been presented with the use of Recursive Neural Networks. The initial result showed that the cross-entropy loss was very minimal for after several epochs. The automatically generated Cebuano sentences also manifested that our language model really worked, though the naturalness of the sentences generated was not yet not perfect but closed to normal sentences. As a conclusion, we can see that there is a huge potential in using RNN in creating a language model given a larger training data.

To further improve the performance of our models, we should be able to use a larger dataset in training our data. Moreover, shuffling the training data might help in providing much complex training data for our model for it to generate much more varied sentences. In doing that, it is possible for us to use Theano to speed up the training of our language model. The application of more complex algorithms such as LSTM/GRU Recurrent Neural Networks can improve the Language Model's performance.

References

[1] B. Karagol-Ayan, "Resource generation from structured documents for low-density languages," Doctoral Dissertation, University of Maryland, 2007.

[2] L. Melnar and C. Liu, "Borrowing Language Resources for Development of Automatic Speech Recognition for Low-and Middle-Density Languages," in Proceedings of the Sixth International Conference on Language Resources and Evaluation, May 2008, pp. 225-230.

[3] A. Ragni, E. Dakin, X. Chen, M. J. Gales, and K. M. Knill, "Multi-Language Neural Network Language Models," in Proceedings of Interspeech 2016, Sept. 2016, pp. 3042-3046.

[4] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," Journal of Machine Learning Research, Vol. 3, pp. 1137-1152, 2003.

[5] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, S. Khudanpur, "Recurrent neural network based language models," in Proceedings of Interspeech 2010, Sept. 2010, pp. 1045-1048.

[6] A. Gandhe, F. Metze, and I. Lane, "Neural network language models for low resource languages," in Proceedings of Interspeech 2014, Sept. 2014, pp. 2615-2619.

[7] E. Arısoy, T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "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, June 2012, pp. 20-28.

[8] G. Attardi, WikiExtractor (Version 2.75) [Python Library]. Retrieved January 25, 2017 from the World Wide Web: https://github.com/attardi/wikiextractor.

[9] D. Britz, "RECURRENT NEURAL NETWORKS TUTORIAL, Part 2 - IMPLEMENTING A RNN WITH PYTHON, NUMPY AND THEANO," [Blog Posts]. Sept. 2015. Retrieved February 23, 2017 from http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-2-implementing-a-language-model-rnn-with-python-numpy-and-theano/

[10] D. Britz, "RECURRENT NEURAL NETWORKS TUTORIAL, Part 1 - INTRODUCTION TO RNNS," [Blog Post] , Sept. 2015. Retrieved February 23, 2017 from http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

[11] T. Mikolov, S. Kombrink, L. Burget, J. Černocký, and S. Khudanpur, "Extensions of recurrent neural network language model," 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), May 2011, pp. 5528-5531.

[12] R. DiPietro, "A Friendly Introduction to Cross-Entropy Loss," [Web log post], May 2016. Retrieved May 30, 2017, from http://rdipietro.github.io/friendly-intro-to-cross-entropy-loss/

[13] M. Mohammadi, R. Mundra, and R. Socher, "CS 224D: Deep Learning for NLP," Lecture Notes: Part IV, Spring 2015. Lecture.

[14] S. R. Bowman, L. Vilnis, O. Vinyals, A.M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," in Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL), August 2016, pp. 10–21.

[15] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM Neural Networks for Language Modeling," In Proceedings of Interspeech 2012, Sept. 2012, pp. 194-197.