Text creation by RNN and LSTM

Anonymous Author(s)
Affiliation
Address
email

Abstract

Text is created with two data set and two algorithm. The yelp data set was used for RNN and Reddit clean jokes data set was used for LSTM. The word2vec was used for word embedding. The word2vec shows good relationship among the data set. The RNN project started to predict a word for users to chose the word automatically. The RNN and LSTM algorithms successfully trained the model and created words. Currently, spelling checking and nltk parser is trying to increase accuracy.

1 Introduction

The final object of this report is to create text by a model which is trained by yelp data set. For this, this project starts by expecting or creating a word based on 4-5 previous words. The text prediction software already exists. T9 is used for auto correct feature in cell phone. Smart Compose is used in Gmail. But all of them are for general purpose.

Up to now, there is no text prediction for specific purpose to write. The model in this report is trained by yelp review data. It will help users to help write a review by recommending words and creating it. To reduce the dimensionality of word, the word to vector is used with around 100 dimension. LSTM model was compared with RNN model to check gramatically correct one.

In this report, a word is predicted based on previous 4-5 words and tried by RNN and LSTM.

1.1 RNN

RNN or Recurrent Neural Networks, as the name suggests, is a repeating neural network[1, 2]. They are the kind whose output from the previous step is fed as input to the current step.

Conceptually they differ from a standard neural network as the standard input in a RNN is a word instead of the entire sample as in the case of a standard neural network. This gives the flexibility for the network to work with varying lengths of sentences, something which cannot be achieved in a standard neural network due to it's fixed structure. It also provides an additional advantage of sharing features learned across different positions of text which can not be obtained in a standard neural network.

A RNN treats each word of a sentence as a separate input occurring at time 't' and uses the activation value at 't-1' also, as an input in addition to the input at time 't'. The diagram below shows a detailed structure of an RNN architecture(Figure 1) .

A RNN treats each word of a sentence as a separate input occurring at time 't' and uses the activation value at 't-1' also, as an input in addition to the input at time 't'. The diagram below shows a detailed structure of an RNN architecture. A RNN treats each word of a sentence as a separate input occurring at time 't' and uses the activation value at 't-1' also, as an input in addition to the input at time 't'. The diagram below shows a detailed structure of an RNN architecture.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

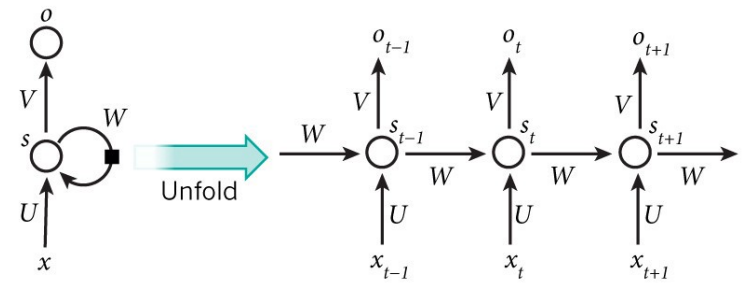

Figure 1: detailed structure of RNN

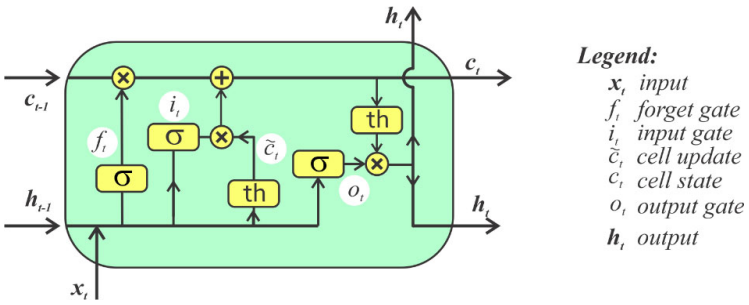

Figure 2: detailed structure of LSTM

1.2 LSTM

Typical RNNs can't memorize long sequences. The effect called "vanishing gradients" happens during the backpropagation phase of the RNN cell network. The gradients of cells that carry information from the start of a sequence goes through matrix multiplications by small numbers and reach close to 0 in long sequences.

The long short term memory cell(LSTM, Figure 2)[3, 4] is an RNN architecture that can memorize long sequences - up to 100 s of elements in a sequence. LSTM has a memory gating mechanism that allows the long term memory to continue flowing into the LSTM cells.

1.3 Word2Vec VS one hot encoding

One-Hot Encoding is a general method that can vectorize any categorical features. It is simple and fast to create and update the vectorization, just add a new entry in the vector with a one for each new category. However, that speed and simplicity also leads to the "curse of dimensionality" by creating a new dimension for each category. Embedding is a method that requires large amounts, both in the total amount of data and repeated occurrences of individual exemplars, and long training time. The result is a dense vector with a fixed, arbitrary number of dimensions. They also differ at the prediction stage a One-Hot Encoding tells you nothing of the semantics of the items. Each vectorization is an orthogonal representation in another dimension. Embeddings will group commonly co-occurring items together in the representation space.

Word2vec(Figure 3)[5] is one of the most popular technique to learn word embeddings using a two-layer neural network. Its input is a text corpus and its output is a set of vectors. Word embedding via word2vec can make natural language computer-readable, then further implementation of mathematical operations on words can be used to detect their similarities. A well-trained set of word vectors will place similar words close to each other in that space. For instance, the words women, men, and human might cluster in one corner, while yellow, red and blue cluster together in another.

There are two main training algorithms for word2vec, one is the continuous bag of words(CBOW), another is called skip-gram. The major difference between these two methods is that CBOW is using context to predict a target word while skip-gram is using a word to predict a target context.

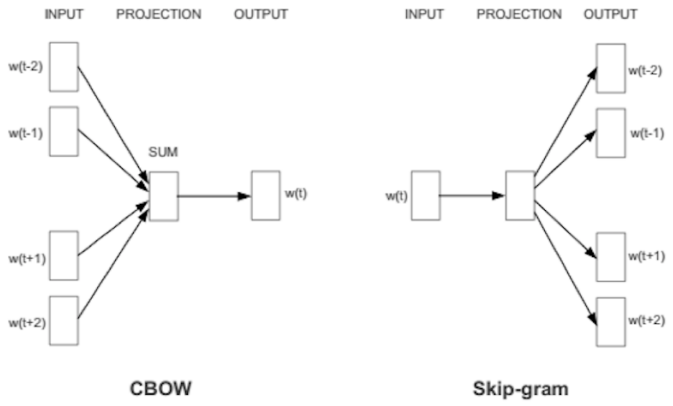

Figure 3: CBOW amd Skip-gram for word2vec

There are two main training algorithms for word2vec, one is the continuous bag of words(CBOW), another is called skip-gram. The major difference between these two methods is that CBOW is using context to predict a target word while skip-gram is using a word to predict a target context. Generally, the skip-gram method can have a better performance compared with CBOW method, for it can capture two semantics for a single word. For instance, it will have two vector representations for Apple, one for the company and another for the fruit(Figure 2).

2 Materials and Method

2.1 Word Processing

The stop word are removed and vectorized by gensim[6]. The Porter stemmer[7] is used for stemming. Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. For example: words such as “Likes”, ”liked”, ”likely” and ”liking” will be reduced to “like” after stemming. The stemmer is based on the idea that the suffixes in the English language are made up of a combination of smaller and simpler suffixes. The main applications of Porter Stemmer include data mining and Information retrieval. However, its applications are only limited to English words. Also, the group of stems is mapped on to the same stem and the output stem is not necessarily a meaningful word.

2.2 Data structure

All sentences in a yelp review are considered as one sentence. The 1st to 4th words are used as a independent variable and 5th word is used as dependant variable. And then 5th to 9th word were used for x variable again and 10th variable become dependent variable. This pattern keep continuing untill it reaches to the end of a review. The next review starts again.

The size of review data is 6 GB. The reviews were randomly selected to reduce the size 1/10, 1/100 and 1/1000. And the results were compared. It is downloaded from https://www.yelp.com/dataset/download .

The Reddit clean jokes dataset is downloaded from https://raw.githubusercontent.com/amoudgl/short-jokes-dataset/master/data/reddit-cleanjokes.csv . The data size is 149k. It has 1623 jokes.

2.3 word2vec

The word2vec was executed by gensim python library for yelp data set. Gensim library enables us to develop word embeddings by training our own word2vec models on a custom corpus either with CBOW of skip-grams algorithms.

The number of dimensions of the embeddings and the default is 100. The number of dimensions of the embeddings was changed to 50 200. The maximum distance between

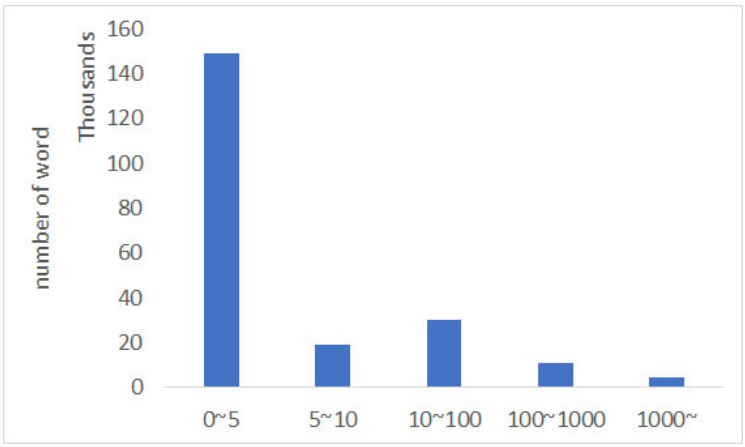

Figure 4: word frequency for yelp data set

a target word and words around the target word. The window size is adjusted to 5 because the length of training sample is 4 . The minimum count of words to consider when training the model; words with occurrence less than this count will be ignored. The default for $\min_c ount is 5. The training algorithm, either CBOW(0) or skipgram(1). The training algorithm is set to skipgram. The$

For Reddit jokes data set, Unique words are calculated in the dataset to define the size of the network's vocabulary and embedding size.

2.4 RNN and LSTM

The vanilla RNN and LSTM in pytorch were used. For the RNN, the hidden dimension is 3 and the number of layers is 1. The loss function is cross entropy. The optimizer is adam.

2.5 Measure the accuracy

The accuracy of predicted word is measured by the average of first 100 similarity between ground truth and predicted words. The similarity is measured is by cosine similarity. It measures the cosine of the angle between two vectors projected in a multi-dimensional space. The cosine similarity captures the angle of the word vectors and not the magnitude. Under cosine similarity, no similarity is expressed as a 90-degree angle while the total similarity of 1 is at a 0-degree angle.

2.6 Computer System

The CPU is AMD ryzen 3900X. The momory size is 32 GB. The GPU is NVIDIA RTX 2060.

3 Results and Disucussion

The corpus of yelp has total 82,578,710 words including repetition. Among them, the 150,000 words has only 0 5 frequency(Figure 4) . Usually, that kind of rare words are misspelling or the words which we can not understand such as 'mwaaa' or 'kammi'(Table 2). To avoid that kind of rare word or misspelling, the predicted word extends as mentioned above.

The Reddit clean jokes data set has total 23914 word including repetition(Figure 5). Around 4.000 words appeared just once. The words with one frequency include punctuation mark. But this data set has no misspelling.

The similarity from word2bec showed good match for words. The kimchi is Korean food. The top 5 similar words of kimchi are all of korean food such as kimchee, bulgogi, jachae and so on(Table 1). It is same as car. The top similar word of car are all the car related words such as vehicle, jeep and suv. However, the persnoal nouns did not show the good similarity. All the similar words with "we" has nothing to do with "we". They are "expesed", "ugrades", "hlm" and so on.

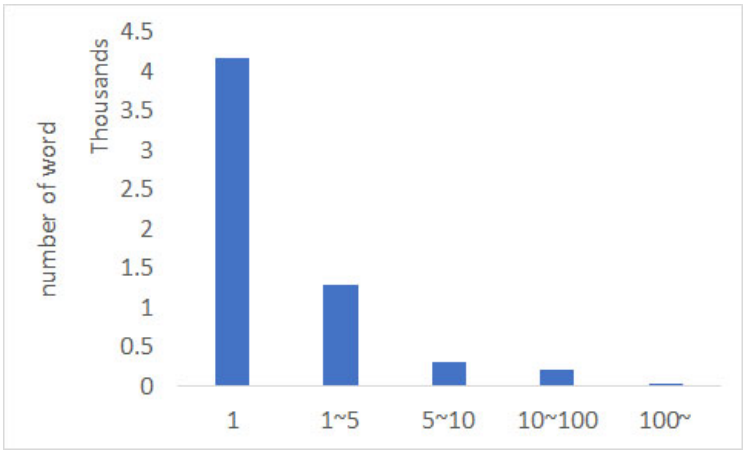

Figure 5: word frequency for Reddit clean jokes dataset

kimchi	korean	apple	car	we
kimchee (0.83)	japanese (0.81)	apples (0.68)	vehicle (0.91)	expensed (0.62)
bulgogi (0.78)	asian (0.76)	crostada (0.66)	jeep (0.8)	ugrades (0.61)
japchae (0.76)	chinese (0.75)	blackberry (0.65)	cars (0.79)	hlm (0.6)
bibimbap (0.75)	filipino (0.75)	crodough (0.65)	vehicles (0.78)	saltlik (0.6)
chigae (0.75)	taiwanese(0.72)	stredudel (0.64)	suv (0.77)	tumbleweeds (0.6)

Table 1: Training

The loss for RNN decreased 18 to 9 after first epoch. It kept decreasing to 7.64 until 500 epoch(Figure 3). The epoch was tried to until 9000 epoch(Figure 5) .

The loss for LSTM decreased 7.15 to 0.45 after 140 epoch. After that, there was no change of loss(Fogure 6).

The average similarity of first 100 data for RNN is 0.48. The similarity is from 0 to 1. If the frequency is less than 10, they are usually misspelling, Spanish or very rare word such as attanpt , antina mwaa and so on. So the expected word is chosen to have closest similarity with the frequency which is greater than 10(Table 2). This algorithm is designed to recommend the predicted word. So the user have freedom to chose a word.

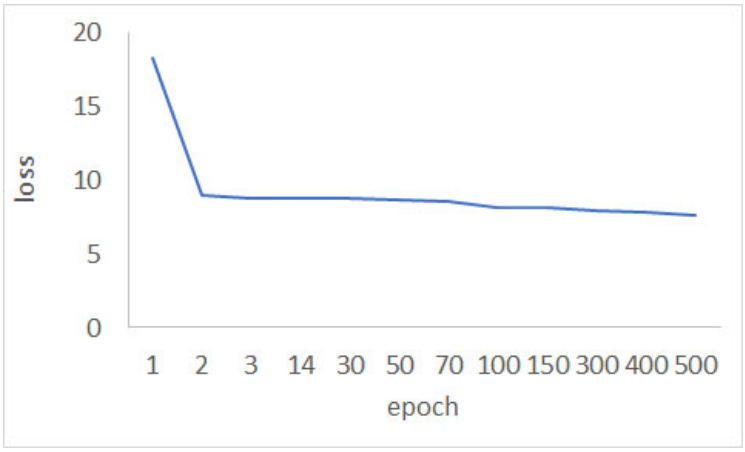

Figure 6: changing loss according to epoch for RNN

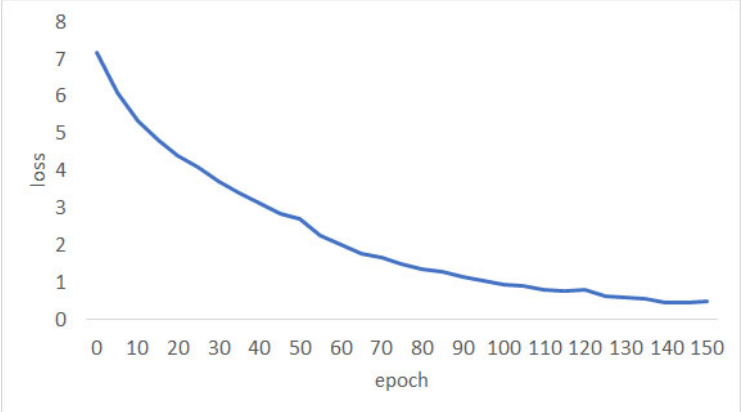

Figure 7: changing loss according to epoch for RNN

given 4 words	5th word	predicted 5th word	frequent 5th word
pasty to satisfy any	taste	attampt (0.52)	zarra
madison can get there	for	antina (0.49)	industri
that going to cost	extra	mwaaa (0.46)	lobsterme
the estimated delivery time	unkonwn	kammi (0.51)	unconvinced
how you like it	or	attampt (0.52)	procrastinate

Table 2: test set and predicted 5th words. The predicted 5th words extends until it meets frequency of 10.

One of unexpected result is that it has frequent 'attampt' as 5th prediction. The similarity with ground truth is 0,45 to 0.52. It seems that the RNN tend to predict the specific words and it happen to be similar with 'attampt'. The 'attampt' is obviously miss spelling. It needs spell check. To avoids such a misspelling or rare words, the predicted results extended to 2000 most similar words. The words whose frequency is at least 10 and meets first is chosen. If there is no such a words, the first 5th frequency is selected. If there is not such a word, the largest frequency among the 2000 words are selected. If the algorithm just select the largest frequent word, it could select irrelevant word.

For the text creation, the first 4 words were given and the algorithm chose the 5th word. And the second words becomes first words and it keep going such a way. Some of result are "the food we want the joons and the jambalya the", "we are convnced that ddnt and redonkulous unbeliev readi tranishing". The 'ddnt' and 'redunkulous' means "never tickle a sleeping dragon; found in Harry Potter book series" and significantly more absurd than ridiculous, to an almost impossible degree", repectively. The other is "apple is good but mediocr the ddnt the somedays the".

I tried other data and alogrithm. The data is Reddit clean jokes data set which has no misspelling. The algorithm is the vanilla LSTM. One of the result is "apple is good but U-turns whenever Savage weigh the hippie joke He was" for the given words "apple is good but".

In both cases, they need to be grammatically correct. I am still working on refining this algorithm. It needs spelling check and pos tagging.

4 further work in the future

We need spell check. The misspelling can be removed or be changed to be correct. The pyspellchecker can be used for it. To make the created text grammatically correct, I am trying to use parser which is in nltk.

References

[1] Dupond, Samuel (2019). "A thorough review on the current advance of neural network structures". Annual Reviews in Control. 14: 200–230.

[2] Abiodun, Oludare Isaac; Jantan, Aman; Omolara, Abiodun Esther; Dada, Kemi Victoria; Mohamed, Nachaat Abdelatif; Arshad, Humaira (2018-11-01). "State-of-the-art in artificial neural network applications: A survey". Heliyon. 4 (11): e00938. doi:10.1016/j.heliyon.2018.e00938. ISSN 2405-8440. PMC 6260436. PMID 30519653.

[3] Fernández, Santiago; Graves, Alex; Schmidhuber, Jürgen (2007). An Application of Recurrent Neural Networks to Discriminative Keyword Spotting. Proceedings of the 17th International Conference on Artificial Neural Networks. ICANN'07. Berlin, Heidelberg: Springer-Verlag. pp. 220–229. ISBN 978-3-540-74693-5.

[4] Schmidhuber, Jürgen (January 2015). "Deep Learning in Neural Networks: An Overview". Neural Networks. 61: 85–117. arXiv:1404.7828. doi:10.1016/j.neunet.2014.09.003. PMID 25462637. S2CID 11715509.

[5] Mikolov, Tomas; et al. (2013). "Efficient Estimation of Word Representations in Vector Space". arXiv:1301.3781 [cs.CL].

[6] Řehůřek, Radim (2011). "Scalability of Semantic Analysis in Natural Language Processing" (PDF). Retrieved 27 January 2015. "my open-source gensim software package that accompanies this thesis"

[7] C.J. van Rijsbergen, S.E. Robertson and M.F. Porter, 1980. New models in probabilistic information retrieval. London: British Library. (British Library Research and Development Report, no. 5587).