International Journal of Psychosocial Rehabilitation, Vol. 24, Issue 06, 2020
ISSN: 1475-7192

Context Prediction Using Neural Networks

[1]Rohith Manda, [2]Deeba K, [3]S Selvakumar

ABSTRACT — *Word-prediction can be thought of as predicting a word, given its context. It is similar to predictive text used by multiple devices. The concept of word-embeddings is ideal for this type of a problem. Embeddings are an alternate form of word representation, such that it preserves the syntactic and semantic information. It extends the functionality to extract or study the relationships between various words, given in a corpus. The objective can be viewed as, (i) generation of quality embeddings,(ii) using neural networks to realise a model, capable of predicting words accurately, given the context as input. The generation of embeddings are a tedious task and requires a good amount of computation. To use these in an environment that is supported by machine learning algorithms requires transformation of these embeddings and corpus into vector space, i.e, a numeric form of representation. Word prediction can be of many types, depending on the model. For example, if the objective is to predict the (n+1)th word, given the previous n words; The automaton can consider only the previous n or (n,n-1) or (n,n-1,n-2…) words for predicting the next word, the earlier two are referred as bigram and trigram model respectively.*

KEY WORDS—*Prediction, Neural Networks.*

I. INTRODUCTION

word embeddings has proven itself to be of great use in a variety of tasks in the domain of NLP such as machine translation, parsing, semantic search, and tracking the meaning of words and concepts over time. Current systems and natural language processing techniques use words as a morphed unit, or an atomic unit. The process of stemming[1] and morphing[1] is quiet common in these techniques. There is no basis for directly using them in machine learning algorithms, unless the are converted into another form of representation. The concept of capturing word similarities was introduced in 2013, after google came out with new techniques to calibrate and calculate this.

In the International Conference on Learning Representations (ICLR)[6], the work of Tom Mikolov revealed that the Skip-gram architecture works slightly worse on the syntactic tasks than the bag-of-words model, however performance metrics show, it is better than NNLM[2].

[1] *Dept. of Computer Science and Engineering,SRM Institute of Science and Technology,rohithmanda_manda@srmuniv.edu.in*

[2] *Dept. of Computer Science and Engineering,SRM Institute of Science and Technology,deebak@srmist.edu.in*

[3] *Dept. of Computer Science and Engineering,SRM Institute of Science and Technology,selvakus@srmist.edu.in*

Received: 22 Sep 2019 | Revised: 13 Oct 2019 | Accepted: 15 Jan 2020

In the same paper,” Distributed Representations of Words and Phrases and their Compositionality”[5], the concept of Negative Sampling Subsampling of Frequent Words was displayed, which showed improved results. This introduction of, methods to learn/train high quality or high density word vectors or embeddings from huge data sets. This methodology would later go on to capture multiple degrees of commonness and similarity between the words in the corpus. The concept is to use similarity cosines of all the word pairs during training/clustering[9] . Unlabeled data then, can be used .

In the work displayed by Wen-tau Yih, Geoffrey Zweig in “Linguistic Regularities in Continuous Space Word Representations”[4], it revealed the usage of Recurrent Neural Network Model and the Vector Offset[2] Method to prevent program stalls and reduce processing time. The only restriction is that documents need to be split into word pairs and that their order should be preserved.
An example could be, Using an offset technique[2] of simple operations of algebraic nature, which are performed on the word embeddings, an example can be, vector(”King”) - vector(”Man”)+ vector(”Woman”) leads to a vector that's nearest to the vector illustration of the word Queen and thus being able to find similarity between twowordsusingtheseembeddings.Once we have the embeddings, their quality are measured using a test , such as a word similarity task. The current results are always compared with the previous ones using neural networks. It is shown that the vectors provide optimal performance on the selected datasets for measuring semantic and syntactic word similarities. The word2vec model in the domain natural language processing, allows us to implement this. Given a set of surrounding words, the target word is predicted using word2ec model.
The input to the model should be In the form wi−2, wi−1, wi+1, wi+2; i.e, the previous and following words of the present word that is selected. The output of the neural network is going to be w_i.

II. WORD2VEC MODEL

Word2vec[6] is a methodology to efficiently produce word embeddings and has been around since 2013.It can be used in recommendation engines and search engines.In this paper, we tend to target distributed representations of words learned by neural networks. Word2vec helps to implement the bag-of-words and the continuous skip-gram model. In the bag-of-words design, the model predicts the center word from it’s surrounding words. The order of word pairs, has no influence on the prediction (bag-of-words assumption).In continuous skip-gram[6], the context word/window is used to predict the surrounding words of the context word. The skip-gram design weighs close context words additional heavily than additional distant context words.

The average word embeddings are trained with word2vec.Default settings used : minimum word frequency 5, word embedding size 300, context window 5, sample threshold 10-5, no hierarchical softmax, 5 negative examples.

III. RELATED WORKS

Neural Networks implementation now ,delivers massive upgrades in performance in many speech, comp vision , natural language processing and machine learning techniques.

Because these trendy NNs usually comprise multiple interconnected layers, this new NN analysis is commonly cited as deep learning.

A number of research papers have shifted their focus on NN[9] approaches to Information Retrieval. The trend to use NNs for IR[9] suggests that the state of affairs can be dynamical. The concept of word embeddings are used here. Models like these, use the newly formed word embeddings as the first information to process and be learned along with all-model parameters.

This is AN exciting direction, as researchers area unit broadly speaking operating toward developing end-to-end NN architectures for IR, which can depart considerably from ancient IR models.

IV. METHODS

A probabilistic feed-forward neural network model is proposed, It consists of input, projection, hidden and output layers. At the input layer, M previous words are encoded using 1-of-Vc coding, where Vc is size of the vocabulary. Input layer is succeeded by a projection layer S that has dimensionality $M \times D$, using a shared projection matrix. The computation of the projection layer is not very costly because of the size M. The architecture becomes very complex when it hits the hidden layer, as the projection layer ,where the context matrix is formed is dense. For an example of $M = 10$, the size of the projection layer (S) might be 400 to 3000, while the hidden layer size H is typically 400 to 2000 units. The hidden layer computes probability functions and graphs over all the words in the vocabulary, resulting in an output layer with dimensionality Vc. Thus, the computational complexity per each training example is $Q = 2*(M\times D)\times H + H \times Vc$,where the dominating term is $H \times Vc$ With binary tree representations of the vocabulary, the number of output units that need to be evaluated can go down to around log2(Vc). Thus, most of the complexity is caused by the term $M\times D\times H$.A hierarchical soft-max model where the vocabulary is represented in terms of a Huffman-tree. Huff-man trees reduces the amount of word-pairs to be checked by assigning binary signatures to the most frequently used ones, while balanced binary tree would require log2(Vc) outputs to be evaluated, the tree from hierarchical soft-max requires only about log2(Unigram perplexity(Vc))[8].The computational speedup of the network changes according to the size of the corpus, for one million words, its value was about 2. Terms have historically been encoded as separate symbols that can not be directly compared to at least one another (unless we have a tendency to take into account edit distance at the character-level).

One-hot encoding is used, in which, each term is shown as a sparse vector, where its dimensionality equals V_c(vocabulary size).

Considering the one-hot vector, we create a zero vector a Neural Language Translation Model (NLTM)[11].They estimate the interpretation likelihood between terms because the trigonometric function similarity of the 2 terms divided by the add of the trigonometric function similarities between the translating term and every one of the terms in the vocabulary.

Previous approaches indicate that NLTM[2] give moderate improvements above MI and classic TM systems. Rather than implementing modest improvements to big number of topics, large differences on a few topics are checked for correction. Hyper-parameters[3] in correspondence to word embeddings imply that the following are not highly impacted by manipulations of embedding dimentionality, window size, object model on baselines, performance, and search correctness.If all terms are equidistant to one another, there is no easy way to recognize similarities among them. This has junction rectifier to the ill-famed vocabulary twin drawback within which associate IR system should acknowledge once distinct however connected terms occur in question and document to perform effective matching.

Summarizing, an efficient neural network architecture for obtaining high-quality word embeddings is shown, directly optimized for sentence representations. It is evaluated on the embeddings produced on multiple datasets, from multiple sources and display the robustness of embeddings in different environments.

V. TRAINING OBJECTIVE

A supervised training environment is created, which predicts sentences which occur next to each other in the training data. Specifically, for a word pair, conditional probability is defined p(si , sj) that displays how similar are the words to each other in the training data. We compute the probability p(si , sj) using a soft-max function[5]:

$$P(y=j \mid \Theta^{(i)}) = \frac{e^{\Theta^{(i)}}}{\sum_{j=0}^{k} e^{\Theta_k^{(i)}}}$$

Alternatively, the naïve bayes probabilistic function can also be used:

$$\underset{\text{Posterior Probability}}{P(c \mid x)} = \frac{\overset{\text{Likelihood}}{P(x \mid c)}\,\overset{\text{Class Prior Probability}}{P(c)}}{\underset{\text{Predictor Prior Probability}}{P(x)}}$$

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

The cosine similarities, which are applied for word pairs,(w_i) and the other calculations are done in the penultimate layer and a soft-max is applied in the last layer to produce the final probability distribution.

VI. EXPERIMENTAL SETUP

To test the efficacy of the embeddings we use multiple test sets. We learn word embeddings from an unlabled corpus. For every sentence pair in the test sets, we compute the conditional probabilities by averaging the word embeddings of each sentence. There are cases, where a word may get no embedding because they are not present in the vocabulary. The cosine similarity between the two word embedding vectors is displayed as a final semantic similarity score/reading. All the Models are experimented on a single dataset, such that the newly generated embeddings can be implemented directly and results across multiple models can be compared. No pre-processing steps or incorporation of the embeddings is done, any surplus steps may act as bias, obscuring our main evaluation purpose, thus the embeddings are directly tested.

- ***Data:***

The 'goodbooks-10k' Kaggle Corpus was used to train word embeddings. This corpus contains 7,04,228 already pre-processed sentences in total, which are made up of The archives containing 10000 XML files, originating from 7,087 unique books.

- ***Baselines:***

Two baselines are used for producing word embeddings in the implementation. The similarity scores between word pairs from the baselines, i.e,, the similarity cosine[10] of all the word-pairs that are generated from the dataset, are calculated. Sub-sampling is performed to highlight words with a higher frequency.

- **Network:**

A feed-forward neural network[10] is used, which is simple to implement and provides faster speeds. The movement of information is limited to one direction, which is forward in this case. They contain no loops or feedbacks. The embeddings size can we set between a range of 100-300 with the fact that a word is only considered for evaluation if it is used at least 5 times in the corpus.

The embeddings are initialized and randomized using normal distribution (with μ = 0.0) and learning rate of σ = 0.01. It is obtained by observing the loss on the training data. Training consists of at least one epoch.

The loss function[3], binary cross-entropy, is used. It is given by:

$$\boxed{-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})}$$

M represents the number of classes, ’y’ is the binary indicator, While ‘C’ is for the correct classification, ‘o’ is for observation.

- Evaluation:
- Input: programmers must ________ to understand.
  Output: ”learn”
- input: about to ________ the idea.
  Output:”study”
- input: to the ______ collapse of.
  Output:”catastrophic”.

VII. RESULTS

Fig.1 represents the relationship between words in terms of embeddings. To compare the standards, of various word vectors; it seems simple to indicate that word Delhi is analogous to India and maybe another countries or cities, using them in advanced tasks to match similarities is not recommended as the expected values are affected by noise/bias. It is observed there can be multiple forms of word similarities, for example, ‘greater’ is analogous to ‘huge’ in the same sense that ‘smaller’ is analogous to ‘tiny’.

For the conflict, of 2 pairs of words with an equivalent relationship, a word was needed to be searched for or calculated, such that the meaning would more or less be similar and the analogy would not change. This can be achieved by performing simple algebraic operations with the embeddings. For example, To find a word that's kind of like little within the same sense as best is similar to good, we can simply compute vector X = vector(”best”)−vector(”good”) +vector(”bad”). Then, the word closest to the vector area of X, is calculated by a circular distance function. When complete training of the word-vectors and embeddings is done, the results will see a drastic improvement.

Table 1: results in terms of Pearson's r, decimal precision of two.

Dataset	**Vector-offset[2]**	**Neural Networks**
Headlines collection	0.56	0.641
Google Books	0.42	0.408
Twitter news	0.63	0.735
News- deft	0.57	0.582

VIII. CONCLUSION

From the results, we can infer that the generated set of word embeddings are of better quality, when neural networks along with Word2Vec model is used, which gives us multiple functionalities to improve on the model, whereas older methods such as vector offset, or bag of words, is statistically less yielding. Neural networks have a lot of potential in terms of efficient implementations of multiple machine learning problems.

The resultant vectors catch minute semantic and syntactic relationships between words, such as a city and the country it belongs to, e.g. Delhi is to London as India is to United Kingdom. Word vectors with such linguistics relationships might be accustomed improve several existing natural language processing applications.

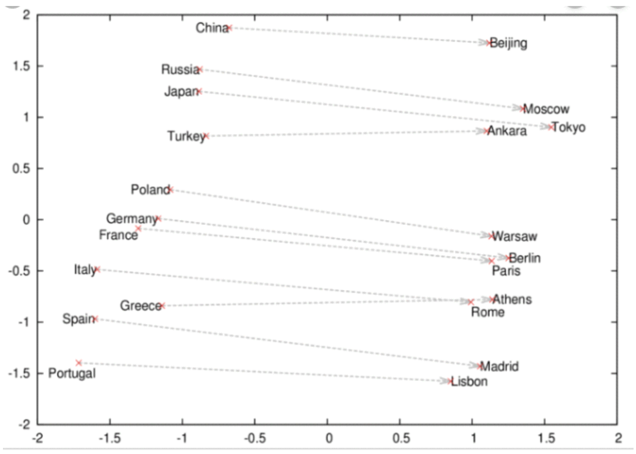
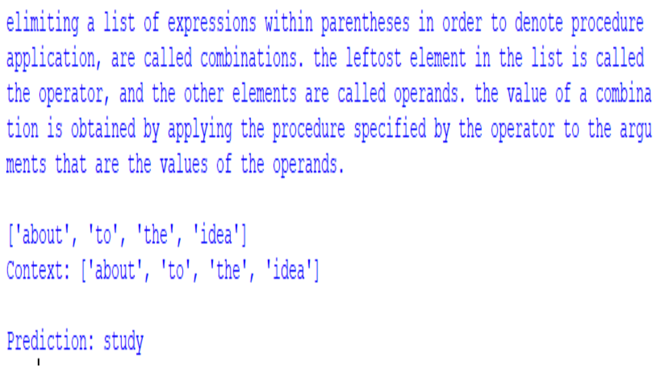


Figure 1: result representation (word similarity)

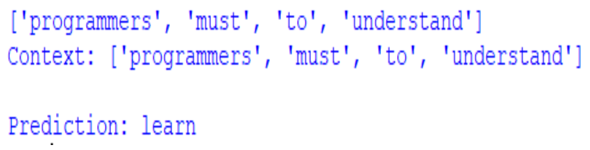


```
elimiting a list of expressions within parentheses in order to denote procedure
application, are called combinations. the leftost element in the list is called
the operator, and the other elements are called operands. the value of a combina
tion is obtained by applying the procedure specified by the operator to the argu
ments that are the values of the operands.

['about', 'to', 'the', 'idea']
Context: ['about', 'to', 'the', 'idea']

Prediction: study
```

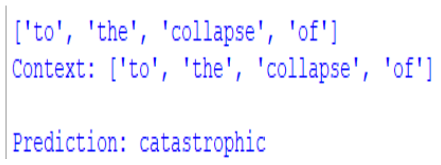
Figure 2: output2



```
['programmers', 'must', 'to', 'understand']
Context: ['programmers', 'must', 'to', 'understand']

Prediction: learn
```

Figure 3: output1



```
['to', 'the', 'collapse', 'of']
Context: ['to', 'the', 'collapse', 'of']

Prediction: catastrophic
```

Figure 4: output3

IX. ACKNOWLEDGEMENTS

I cannot express enough thanks to, my parents,my mentor, Ms. Deeba.K Ma'am, for their support . The department of computer science and engineering, SRM institute of science and technology, has also displayed immense support and encouragement from their side. I am grateful to everyone, who provided me with the learning opportunities which gave me a chance to prove myself.

The chance to work on a project like this, has taught me a lot about the field and the protocols one must follow to pursue research work.

REFERENCES

1. Y. Bengio, R. Ducharme, P. Vincent. A neural probabilistic language model. Journal of Machine Learning Research, 3:1137-1155, 2003.
2. Y. Bengio, Y. LeCun. Scaling learning algorithms towards AI. In: Large-Scale Kernel Machines, MIT Press, 2007.

3. Sebastian Sudholt, Gernot A. Fink “Evaluating Word String Embeddings and Loss Functions for NN-Based Word Spotting”. IAPR International Conference on Document Analysis and Recognition (ICDAR) 2017.
4. Wen-tau Yih, Geoffrey Zweig “Linguistic Regularities in Continuous Space Word Representations”. North American Chapter of the Association for Computational Linguistics(NAACL) 2014.
5. Zhila, W.T. Yih, C. Meek, G. Zweig, T. Mikolov. Combining Heterogeneous Models for Measuring Relational Similarity. NAACL HLT 2013.
6. Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean” Efficient Estimation of Word Representations in Vector Space”. The International Conference on Learning Representations (ICLR) 2013.
7. D.A. Jurgens, S.M. Mohammad, P.D. Turney, K.J. Holyoak. Semeval-2012 task 2: Measuring degrees of relational similarity. In: Proceedings of the 6th International Workshop on Semantic Evaluation (SemEval 2012), 2012.
8. A.L. Maas, R.E. Daly, P.T. Pham, D. Huang, A.Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In Proceedings of ACL, 2011.
9. T. Mikolov, A. Deoras, D. Povey, L. Burget, J. Cernock ˇ y. Strategies for Training Large Scale ´ Neural Network Language Models, In: Proc. Automatic Speech Recognition and Understanding, 2011.
10. T. Mikolov, W.T. Yih, G. Zweig. Linguistic Regularities in Continuous Space Word Representations. NAACL HLT 2013. H. Schwenk. Continuous space language models. Computer Speech and Language, vol. 21, 2007.