Finding semantic similarity among words using Word2vec model in deep neural network.

Submitted by : Md Maminur Islam
Email : mislam3@memphis.edu

Introduction : Neural Network is being popular day by day and it's being applied in wide range of area like speech recognition, image recognition, natural language processing etc. Finding semantic similarity among words is a vital part in natural language processing and there are various models available in deep network to find the similarities among words. In my project, I implemented a deep model named Word2Vec which finds semantic similarity among words.

Word2Vec is a group of deep models that implements word embeddings and can be used to find similarities among words. Word2vec model was first created by a group of researchers at google led by Tomas Mikolov and subsequently analysed and explained by other researchers and it has many advantages over other earlier algorithms like latent semantic algorithm.

Word2Vec is a very simple model having only input layer, hidden layer and output layer. There are many complex model available in neural network which could be applied for finding the semantic similarity among words. However, the performance of complex models degrades especially the models become slow as the size of corpus data increases. The main intuition behind the Word2Vec was - achieve better performance not by using a more complex model but by using a simpler model to be trained on much larger amounts of data.

Word2vec implements two types of model - continuous-bag-of-words(CBOW) and skip-gram. In my project, I implemented skip-gram model for word2vec which predicts similar words for a any word. To train the model, I crawled texts from 10,000 documents which produced tons of words and after removing the stop words, my ultimate goal is to find neighbouring words for important terms from training data.

Tensorflow is a popular deep models framework and genism is developed on top of the tensorflow which has a implementation of word2vec skip-gram model in python. I used genism in my project to implement the word2vec. I used Amazon EC2 instance to implement my project where I the environment ready to run tensorflow with other required frameworks. To obtain the training data, I also implemented crawler in python which collected 10,000 documents from University of Memphis website.

Related Works:
A numerous number of research is going on learning semantic similarity among words. There are several popular approaches and WordNet is the one of the old models for calculating semantic similarity which was later mostly replaced by Wiktionary. WordNet is not that much effective as it requires expert linguists which is not feasible. Wiktionary introduced the idea of exploiting semantic similarity from the users from web rather than linguistic experts and is available for multiple languages. Although Wiktionary schema is fixed and not enforced, but older entries are not updated. Hence, this is practically inconsistent. Word co-occurrence

method is another popular method in information retrieval for calculating semantic similarity where every query is considered as a document and corpus documents as well as query documents are represented as vectors and the relevant documents are retrieved based on vector similarity between query vector and document vectors. However, this method has some drawback. Firstly, it ignores the ordering of words on documents. Secondly, It does not consider the contextual meaning of the words in a sentence. Another popular approach is lexical database methodology which has a predefined words hierarchy with words, meaning and relationship with other words which is store in a tree like structure. However, appropriate meaning of the words is typically not considered at the time of calculating similarity. Moreover, meaning of words varies from document to document. In edition, all the mentioned methods as well as other models requires feature engineering and significant efforts are required to work with any other languages. But Word2Vec in neural network requires no such effort and document corpora can be directly feed to the model in determining semantic similarity. Hence, this model works with any other languages as well without significant effort.

Approach:
Word2vec is a simple model having one input layer, one hidden layer and one output layer. The input to the model is documents from corpora. But we can not directly input documents the model. Some preprocessing on documents are required for that. Firstly, we make words pairs from the corpora. The amount of word pairs depends on window size as well. For each sentence in a document, we start at the beginning of the sentence and set the first word of the sentence as first word of the pairs. The other word of the sentence is each of the words to left and to right of the first word in sentence within the limit of window size. Then we slide the window to right and we add word pairs again. Repeating the above process, we complete generating word pairs for the whole corpura.

The next task is to convert word pairs to vector. Suppose, our vocabulary has 10,000 words. Hence, we represent the whole vocabulary by 10,000 x 10,000 matrix where each row in the matrix is a vector. Each word has a position to the vocabulary. We set that position in the vector to 1 and other positions are set to 0. So, each word is represented by a vector of size 10,000 where only 1 position contains 1 and other positions contain 0. We generate one hot vector for each word this way. Hence, we later replace the word pairs by vectors. First vector for each pair from the vector pairs is the input vector and second vector is the related vector which is given as input to the Word2Ve model. Hence, the input layer in the Word2Vec model has 10,000 neurons and the hidden layer can contain arbitrary number of neurons which is typically 300 and the output layer also contains 10,000 neurons. The Word2Vec model first computes the weight matrix values in the hidden layer. The ultimate goal of training this model is to learn the weight matrix. When the learning or training is complete, we can compute similarity among words. For the skip-gram model, input is a word and the output is probability for other words in the vocabulary which represents the similarity between input word and each of the words from vocabulary. For continuous-bag-of-words(CBOW) model, the input is a sentence with a missing word which is a list of words and the output is the most appropriate word as the missing word.

Implementation:
Several libraries available as a implementation of Word2Vec model. I developed my project in python. Genism in python implements Word2Vec model where I needed to only provide the sentences from the corpora, window size and how many similar words I want to see. Hence, firstly I crawled 10,000 documents from university of memphis website. From the documents, I removed the stopwords. Later, I generated sentences out of the corpora which I directly feeded as input to the model. I fixed the window size to 5 and I wanted to find out only 10 most similar words. I implemented skip-gram model in python.

Experiment results

Word	Similar word 1	Similar word 2	Similar word 3	Similar word 4	Similar word 5	Similar word 6	Similar word 7
International	Readmission	Law	Students	Freshman	Undergrads	Started	Graduate
Admission	Applicants	Employes	Review	Requirements	Decision	Against	Minimum
Finance	Business	Career	Estate	Premier	Source	Banking	Kyle
Insurance	Plan	Addiction	Affordable	Highlight	Tuition	Managed	Risk
Recreation	Sports	Intramural	Athletic	Club	Scene	Echles	Games
Students	Faculty	Readmission	prospective	current	members	Admissions	parents
Computer	Kumar	Lab	Rom	Stored	Santosh	Science	Laptop
Data	Big	Analytics	Tools	Knowledge	Platform	Retrieval	Collect
Language	Speech	Pathologists	Pathology	Ital	Foreign	Audiology	Chinese

Table: Experiment result showing 10 most similar words for 10 input words

Future Works
There is another popular method for calculating semantic similarity among words name GloVe. In future, I want to implement that method and want to compare the results with Word2Vec. Moreover, I also will try to combine two methods so that I can improve the performance. I only applied Word2Vec on 10,000 documents from university of memphis website. But in future, I want crawl documents from internet specially from social media and want to apply both method.

Conclusion
From the table of experiment result, we see that Word2Vec showed very good performance in finding similar words from the University Of Memphis website. However, I only trained on 10,000 documents. But the major advantage of Word2Vec is that we can train this model on large document corpus and the bigger the training data, the better the performance. So, if I could train

the model on more documents from the university website, hopefully the performance would have been better.

Hence, Word2Vec is a very effective model in finding similar contextual words and it is already being used by several tasks in natural language processing i.e. machine translation, sentiment analysis, information retrieval, speech recognition, question answering etc. and it is being widely used because of it’s better performance although it is a very simple mode.

References

1. https://www.tensorflow.org/tutorials/word2vec
2. http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/
3. https://rare-technologies.com/word2vec-tutorial/
4. Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on, pages 5528–5531. IEEE, 2011.
5. ‘Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. arXiv preprint arXiv:1206.6426, 2012
6. Tomas Mikolov, Anoop Deoras, Daniel Povey, Lukas Burget and Jan Cernocky. Strategies for Training Large Scale Neural Network Language Models. In Proc. Automatic Speech Recognition and Understanding, 2011.
7. Tomas Mikolov. Statistical Language Models Based on Neural Networks. PhD thesis, PhD Thesis, Brno University of Technology, 2012
8. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. ICLR Workshop, 2013
9. Peter D. Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. In Journal of Artificial Intelligence Research, 37:141-188, 2010.
10. Jason Weston, Samy Bengio, and Nicolas Usunier. Wsabie: Scaling up to large vocabulary image annotation. In Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three, pages 2764–2770. AAAI Press, 2011.