Using Word2Vec to process big text data

Poster · October 2015

DOI: 10.1109/BigData.2015.7364114

2 authors:

Long Ma
Troy University
12 PUBLICATIONS **102** CITATIONS
SEE PROFILE

Yanqing Zhang
Georgia State University
47 PUBLICATIONS **1,645** CITATIONS
SEE PROFILE

Using Word2Vec to Process Big Text Data

Long Ma
Computer Science Department
Georgia State University
Atlanta, Georgia
lma5@student.gsu.edu

Yanqing Zhang
Computer Science Department
Georgia State University
Atlanta, Georgia
yzhang@gsu.edu

***Abstract*—Big data is a broad data set that has been used in many fields. To process huge data set is a time consuming work, not only due to its big volume of data size, but also because data type and structure can be different and complex. Currently, many data mining and machine learning technique are being applied to deal with big data problem; some of them can construct a good learning algorithm in terms of lots of training example. However, considering the data dimension, it will be more efficient if learning algorithm is capable of selecting useful features or decreasing the feature dimension. Word2Vec, proposed and supported by Google, is not an individual algorithm, but it consists of two learning models, Continuous Bag of Words (CBOW) and Skip-gram. By feeding text data into one of learning models, Word2Vec outputs word vectors that can be represented as a large piece of text or even the entire article. In our work, we first training the data via Word2Vec model and evaluated the word similarity. In addition, we clustering the similar words together and use the generated clusters to fit into a new data dimension so that the data dimension is decreased.**

Keywords-big data; Word2Vec; clustering; feature reduction

I. INTRODUCTION

Big data [1] is a huge data set with complex structure that has been generated in various fields such as biology, medicine, social networks, and humanities. It is definitely a challenge to handle such a big data set using traditional data processing tools. Some state-of-art machine learning algorithms and statistical learning strategies have been applied to big data mining; many of them can create good learning algorithms by training a large-scale data set. Unfortunately, the corresponding computing cost is much higher than dealing with smaller data set. In order to make learning algorithm more efficient and eliminate the risk of over-fitting, we are seeking for a way to decrease the big data dimension. Word2vec [2], proposed by Google, is a neural network that processes the text data. Word2Vec is not a single algorithm, but it includes two learning models, Continuous Bag of Words (CBOW) and Skip-gram [3,4]. CBOW predicts the word given its context, but Skip-gram predicts the context given a word. Through feeding the text corpus into one learning model, Word2Vec finally generates the word vectors. In this process, Word2Vec firstly builds a vocabulary from training text corpus and learns the vector representations of each word. In addition, Word2Vec has an ability to calculate the cosine distance among each word. Therefore, we have benefit to clustering the similar words based on their distances. By grouping the similar words, original feature dimension is projected to a new lower dimension. In our work, we first input large-scale text corpus to Word2Vec to produce word vectors. In the next step, an unsupervised machine learning algorithm (*K*-means) [5] is used to clustering those word vectors. Given the value of *K*, the words (features) are classified into *K* distinct clusters, thus the feature dimension is reduced to *K*. In our research, the amount of training instances is less than the collection of features, so that the Word2Vec provides a good strategy to decrease the feature dimension, not only for time cost but also on classification task. In the multi-class classification task, we implement one of popular machine learning algorithm on training data and evaluate its classification performance via 10-fold cross-validation.

In this paper, we introduce the Word2Vec and evaluate its learning model in word similarity task in the second section. In the third section, we use *K*-means clustering algorithm to group similar words in order to decrease the feature dimension. In the next section, we apply a multi-class classification algorithm to evaluate the classification performance. We finally conclude our work in the last section.

II. WORD VECTOR GENERATION

The training data we use is one popular data set, the 20 Newsgroups data set [6], which was originally collected by Ken Lang. The data is organized and split into 20 newsgroups, and each newsgroup corresponds to the different news domain, such as, *sci.med* and *rec.autos*. We only use its 11,314 training data to evaluate our model. Although the training data size is not too large, it should be a good example to introduce our model. Before inputting the training data to Word2Vec, we only remove punctuations for each training instance. We apply the gensim [7], which is a python library to help us implement the Word2Vec. In this process, we first build a vocabulary from the entire training data. In order to generate the word vectors well, we employ the Skip-gram model because it has a better learning ability than CBOW if we are not taking computing speed into account [2]. After training, each word attaches a vector. Finally, we construct a high dimension matrix. Each row in matrix represents every training example and the columns are the generated word vectors. As a consequence, the word has multiple degrees of similarity, it can be computed via a linear calculation. For example, vector ("*Beijing*") – vector ("*China*") + vector ("*America*")

TABLE I. WORD SIMILARITY

Words	Words Similarity			
	1st Word	*Cosine Distance*	*2nd Word*	*Cosine Distance*
computer	evaluation	0.9325	algorithm	0.9292
president	property	0.9306	population	0.9123
american	Churches	0.9438	Greece	0.9365

produces a vector that represents the word "*Washington*". To test the word similarity, we have few examples in Table I, but we only list the two most similar words and their cosine distances. For example, given a word "computer", we obtain the two similar words "*evaluation*" and "*algorithm*" with their distances to "*computer*". The results show the Word2Vec successful find the semantic related words.

III. FEATURE REDUCTION

A. Word Clusterring

In Natural Language Process (NLP), it might achieve better performance by grouping similar words together [8]. To achieve this, we need to look for the centers of each word cluster. Currently, some clustering algorithms can implement this work. Considering the simplicity, we deploy the *K*-means clustering algorithm. *K*-means clustering is a vector quantization method [9], which partitions *N* items into *K* clusters according to the nearest mean calculation. Besides, the distance calculation in *K*-means refers to the Euclidean distance [10]. The only parameter we need to set is the value of *K*; given the different *K* values, we will have *K* different number of clusters. Therefore, all words are grouped into *K* clusters and each word attaches its index of cluster. Table II shows few examples that the contents of *4* clusters when *K* equals to *500*. We can find the similar words are almost correctly grouped into corresponding clusters.

B. Feaure Reduction

The great motivation for introducing feature reduction in our model is that the number of instances (*11,314*) is much less than the amount of original features (*61,189*). Because original words have been grouped into *K* clusters, thus the original feature dimension (*61,189*) is projected to a new feature dimension (*K*). This strategy works similar to PCA [9], and the processed data dimension will be *11,314 * K* (*K << 61,189*). At last, we will use the dimension-reduced data in multi-class classification experiment and values of *K* that we choose are *500*, *1000*, *1500*, and *2000*.

TABLE II. CLUSTER CONTENT

Cluster number	Cluster Contents
1	u'my', u'has', u'is', u'in', u'the', u'had', u'for'…
2	u'components', u'speed', u'trigger', u'applications', u'stage', u'developers', u'manufacturers …
3	u'Medical, u'cigarettes', u'usma', u'smoked', u'food', u'Laboratory', u'chewing…
4	u'population', u'federal', u'laws', u'treasure', u'crime, u'organizations', u'Pope'…

IV. MULTI-CLASS CLASSIFICATION

In order to evaluate our model's performance, we fit our data into multi-class classification task. First, we convert the multiclass classification into multiple binary classifications. Here, we apply One-vs-Rest [11] technique that is training a single classifier for each class. The instance of one class is viewed as positive class; the others are considered as negative. We then apply the LinearSVC (Linear Support Vector Classifier) as a classifier and it is based on LIBLINEAR [12].

V. PERFORMANCE

To evaluate classification performance, we calculate the F1-micro score and record the running time. In addition, we implement the 10-fold cross validation on training data. Table III shows the performance of multi-class classification on dimension-reduced data sets. The first row shows the F1-micro score and time consuming without applying Word2Vec and clustering algorithm. The rest of rows present the dimension-reduced data classification performance. We can find the performances are a little better than the original data set, but the time cost has a relatively big improvement.

VI. CONCLUSION

In our work, we apply the Word2Vec technique into big data processing. Considering the huge data dimension issue when dealing with large-scale training data, Word2Vec provides a way to clustering the similar data. This strategy can be used to reduce the data dimension. To achieve this, we employ two methods. On one hand, we feed our training data into Word2Vec to generate the word vectors. Through applying the linear calculation on each word vector, we can find semantic related words. On the other hand, we group similar words together using *K*-means clustering algorithm. By given values to *K*, we construct *K* clusters. Thus, instead of creating word vectors via vocabulary, we make word vectors based on contents that in each cluster. This strategy decreases the data dimension and speeds up the multi-class classification. The new data dimension lowers the time cost

TABLE III. CLASSIFICATION PERFORMANCE

Data Dimension	Classification Performance	
	F1-Micro Score	*Time Cost (s)*
11,314 * 61,189	*0.7524*	*5 * 10^3*
11,314 * 2000	*0.774*	*8 *10^2*
11,314 * 1500	*0.7619*	*6 * 10^2*
11,314 * 1000	*0.753*	*4 * 10^2*
11,314 * 500	*0.7506*	*3 * 10^2*

in the machine learning task without affecting learning ability from the training data, or even improve learning ability. Therefore, our model will be helpful when dealing with big data.

REFERENCES

[1] M. Viktor, K. Cukier. Big data: A revolution that will transform how we live, work, and think. Houghton Mifflin Harcourt, 2013.

[2] T. Mikolov, K. Chen, G. Corrado, J. Dean, “Efficient Estimation of Word Representations in Vector Space,” Proc. Workshop at ICLR, 2013.

[3] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, “Distributed Representations of Words and Phrases and their Compositionality,” Proc. NIPS, 2013.

[4] T. Mikolov, W.Yih, G. Zweig, “Linguistic Regularities in Continuous Space Word Representations,” Proc. NAACL HLT, 2013.

[5] J. A Hartigan, M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," Applied statistics, 1979, pp.100-108.

[6] K. Lang, “20 newsgroup data set”. Available at: qwone.com/~jason/20Newsgroups/. [Accessed 30-Sep-2015]

[7] R. eh ek, P. Sojka, Proceedings of the LREC 2010 Workshop on New Challenges for NLP Framework. ELRA, Valletta, Malta, 2010, pp.45-50.

[8] D. Ravichandran, P. Pantel, E. Hovy, "Randomized algorithms and nlp: using locality sensitive hash function for high speed noun clustering," Proc. The 43rd Annual Meeting on Association for Computational Linguistics, 2005, pp.622-629.

[9] G. Allen, R. M. Gray, “Vector quantization and signal compression,” Springer Science & Business Media, vol. 159, , 2012.

[10] S. Wold, K. Esbensen, and P. Geladi, “Principal component analysis,” Chemometrics and intelligent laboratory systems, vol. 2, no. 1, 1987, pp. 37-52.

[11] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, 2011, pp. 2825-2830.

[12] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin, “LIBLINEAR: A library for large linear classification,” Journal of Machine Learning Research 9, 2008, pp.1871-1874.