Aicyber's System for IALP 2016 Shared Task: Character-enhanced Word Vectors and Boosted Neural Networks

Steven Du, Xi Zhang
www.aicyber.com
No. 39 Nanjing Road, Capitaland International Trade Center Block B 1604
Tianjin China
{steven,zhangxi}@aicyber.com

***Abstract*—This paper introduces Aicyber's system for IALP 2016 shared task, the Dimensional Sentiment Analysis of Chinese Words. The system is an ensemble of several boosted one layer neural networks, each one is trained on a different type of Chinese word vector. Our best system mainly use position-based character-enhanced word embedding and FastText as word vectors and achieve Mean Absolute Error 0.577(1st) in valence prediction and Pearson Correlation Coefficient 0.671(1st) in arousal prediction.**

***Keywords*-boosting;neural network;word embedding**

I. Introduction

The purpose of this shared task[1] is to predict a given traditional Chinese word's affective states in continuous numerical values (from 1 to 9) on valence-arousal space [1]. It is a supervised learning task, 1653 human labeled training dataset is given, participants are asked to predict valence and arousal value of 1149 words in the test set. System performance is measured by Mean Absolute Error (MAE) and Pearson Correlation Coefficient (PCC).

This paper will present Aicyber's system, from word embedding features, regression model, evaluation towards final submission.

II. Word Embeddings

The training and testing data are all traditional Chinese words. A better representation is required to cover the semantic aspects of them. Several methods could learn such representation from large text corpus. For example the Latent Semantic Analysis [2], Latent Dirichlet Allocation [3], Vector Space Model [4], Explicit Semantic Analysis [5] and the distributed word representation, which is also known as word2vec embedding [6]. In this paper, we use two set of word embeddings derived from the word2vec approach.

A. Character-enhanced Word Embedding

The first set of word embedding is character-enhanced word embedding [7] (CWE). Their work shows semantic meaning of a word is also related its composing characters. The effectiveness of new word vector is confirmed by evaluating on word relatedness computation and analogical reasoning. Two type of embeddings in CWE, the position-based character embeddings (CWE+P) and cluster-based character embeddings (CWE+L) are evaluated, but only CWE+P is included in submitted system due to a bug in script.

These CWE models are trained with window size of 5, 5 iterations, 5 negative examples, minimum word count of 5, Skip-Gram [6] with starting learning rate of 0.025 , the output word vectors are of 300 dimensions.

B. FastText Word Embedding

Similar to CWE, [8] proposes enriching word vectors with sub-word information. Where a word vector is associated to each character n-grams. The open sourced toolkit for their model is known as FastText[2]. It's the second embedding used in submitted system.

FastText word embedding is trained with same setting as CWE training. Please noticed that default minimum character n-gram for English is 3, we set it to 1 for Chinese.

C. Data for Word Embedding Training

We don't have any traditional Chinese text corpus, so word embedding is trained from simplified Chinese.

Following public datasets are used:

1) Chinese Wikipedia Dumps (Time stamp: 2011-02-05T03:58:02Z), however use of the latest dumps[3] is encouraged.
2) Douban movie review[4].
3) Aicyber synthesized 200 sentences[5]. These are intended to cover the out of vocabulary words.

Embedding trained on these datasets contains 445662 word vectors. Word not frequently used in mainland China is mapped to its synonyms in simplified Chinese, for example 强暴犯(Rapists) is mapped to 强奸犯(Rapists). After mapping, most of the words in this task are in vocabulary, except 厂厂. (Which indeed means hehe, an onomatopoeia, we noticed this after made submission.)

III. Regression Model

In submitted system, boosted neural networks is adopted as regression model. It is a boosting system applied on neural networks. Both neural network and boosting method are from Scikit-learn [9], an awesome machine learning package.

[1] http://nlp.innobic.yzu.edu.tw/tasks/dsa_w/

[2] https://github.com/facebookresearch/fastText

[3] https://dumps.wikimedia.org/zhwiki/

[4] http://www.datatang.com/data/45075

[5] https://github.com/StevenLOL/ialp2016_Shared_Task/blob/master/data/extend.txt

A. Neural Network

The neural network used in this work is very simple. It has only one hidden layer, and its size is 100, with $relu$ [10] activation function, $adam$ [11] as its training algorithm and a constant learning rate of 0.001.

B. Boosted Neural Networks

Boosting is a machine learning ensemble strategy to build a committee of learners that may be superior to a single learner [12], [13]. In this experiment we propose use neural network as the fundamental building blocks, so named as Boosted Neural Networks (BNN). The selected implementation of boosting algorithm is AdaBoost.R2 [13].

IV. Evaluation

Evaluation is conducted locally on valence estimation only, the metric used is MAE, it is measured by mean value of 3 rounds of 10 folds cross-validation, each round has constant and unique random seed.

First, we evaluate the performance of different word embeddings then we fix the word embedding type and evaluate different regression methods.

A. Evaluation of Word Embeddings

Besides three embeddings (CWP+P,CWP+L and FastText) mentioned in Section II, we extend evaluation to include word2vec as the baseline feature. Two regression methods, Linear Support Vector Regression [14], [15](LSVR, the baseline) and proposed BNN are used to evaluate the goodness of word embedding features.

This work mainly explores the size of word feature vector and two training schema, the continuous bag-of-words model (CBOW) [6] and Skip-Gram of different type of word embeddings.

In Table I, a LSVR is applied to different type of embedding features, grouped by training methods and size of feature vector. **C** denotes CBOW, **S** denotes Skip-Gram, **100/300** denotes embedding size is 100 or 300. Regression target values ranged from 1 to 9. Reported MAE(the lower the better) is an average of 3 rounds of 10 folds cross-validation on training set. Table shows that in all the cases, Skip-Gram has better performance than CBOW. Higher dimension in Skip-Gram training leads to reduction in prediction error, but it doesn't guarantee better MAE for CBOW training.

This is confirmed with Table II, where the proposed boosted neural network is applied on embedding features. It is also clearly seen that new word vectors are better than baseline word2vec when BNN is applied, and the best features are embeddings trained with Skip-Gram of 300 dimensions (S-300). Next we will exam different regression methods based on S-300.

Valence MAE by LSVR Baseline				
Embeddings	**C-100**	**S-100**	**C-300**	**S-300**
word2vec	0.936	0.839	0.950	**0.819**
CWE+P	0.940	0.773	0.940	**0.765**
CWE+L	0.953	0.827	0.923	**0.769**
FastText	1.093	0.796	1.343	**0.765**

Table I

A LSVR is applied to different type of embedding features, grouped by training methods and size of feature vector.

Valence MAE by Boosted Neural Network				
Embeddings	**C-100**	**S-100**	**C-300**	**S-300**
word2vec	0.878	0.757	0.952	**0.756**
CWE+P	0.823	0.702	0.837	**0.670**
CWE+L	0.823	0.741	0.816	**0.662**
FastText	0.876	0.695	0.947	**0.668**

Table II

Boosted neural network regression method applied to different type of features.

B. Evaluation of Regression Models

This section introduces the regression models, training schema, evaluation results and issue with training.

Since we use the boosted neural network for this task. To evaluate boosting effect, the performance of ordinary neural work (NN) is required to be examined. Meanwhile two regressors from boosting family are worth to be mentioned, gradient boosting machine [16], [17] (GBM) and the eXtreme Gradient Boosting [18] (XGB).

During 10 fold cross-validation, regression methods are not trained in the same way. LSVR and GB are trained on 90% of training data. XGB, NN, BNN further divided this 90% of training data into 90% training and 10% validation set, their training will stop once there is no improvement obtained on validation set, this strategy is known as early-stopping.

Table III presents the results of above regression methods along with LSVR baseline. It's quite obvious that BNN obtain better MAE than ordinary NN, both BNN and NN are better than baseline LSVR. GBM and XGB didn't yield superior results to baseline method. We will further investigate the performance of XGB as it is a well known reliable large-scale tree boosting system [19], ruling leader-board of many machine learning competitions [20].

One problem with BNN is that it has the longest running time, to make it run faster, a Principal Component Analysis [21] (PCA) is applied. This reduces the dimensions of word embedding from 300 to 100, with a trade-off in MAE. As demonstrated in Table IV, after PCA, word2vec and FastText have better MAE, but CWE+P and CWE+L are degraded. We found that normalize target value by removing mean and scaling to unit variance (BNN_Norm), will lead to notable improvement for both MAE and PCC.

Through evaluation, the best feature and regression methods are found. The final submission Run1 is an average of BNN_Norm applied on CWE+P and FastText, if CWE+L is included in the ensemble, better result could achieve. Run2 is generated by BNN_Norm on CWE+P feature alone.

C. Discussion

For Run1 submission the MAE obtained on valence testing set is 0.577. It's even better than 3 out of 5

Embedding S-300	Valence MAE by Different Regression Methods				
	LSVR	**GBM**	**XGB**	**NN**	**BNN**
word2vec	0.819	0.829	0.881	0.801	**0.756**
CWE+P	0.765	0.757	0.809	0.729	**0.670**
CWE+L	0.769	0.795	0.860	0.730	**0.662**
FastText	0.765	0.791	0.847	0.711	**0.668**

Table III
EVALUATION OF DIFFERENT REGRESSION METHODS APPLIED TO S-300 EMBEDDING.

Embedding S-300-PCA100	Valence MAE		Valence PCC	
	BNN	BNN_Norm	BNN	BNN_Norm
word2vec	0.702	**0.686**	0.858	**0.867**
CWE+P	0.678	**0.623**	0.874	**0.891**
CWE+L	0.671	**0.627**	0.873	**0.891**
FastText	0.662	**0.639**	0.879	**0.889**

Table IV
NOTABLE IMPROVEMENT IN MAE AND PCC MADE BY NORMALIZED BNN APPROACH.

human annotators mentioned in [1]. But the PCC value is only 0.848, it is not as good as expected. We are still investigating this problem. Anyone interested in solving this problem or try to reproduce our results could refer to our public repository[6].

V. CONCLUSION

This system paper had covered Aicyber's system for Dimensional Sentiment Analysis of Chinese Words. During feature evaluation, CWE+P, CWE+L and FastText word-embedding are identified as the best features. Then local cross-validation also demonstrated the effectiveness of BNN. To tackle training speed problem, PCA is applied on embeddings. The target normalization further improved system performance. Submitted system achieved MAE 0.577(1st) in valence estimation and PCC 0.671(1st) in arousal estimation.

REFERENCES

[1] Liang Chih Yu, Lung Hao Lee, Shuai Hao, Jin Wang, Yunchao He, Jun Hu, K Robert Lai, and Xuejie Zhang. 2016. Building chinese affective resources in valence-arousal dimensions. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

[2] Landauer, T. K.; McNamara, D. S.; Dennis, S.; and Kintsch, W. 2007. Handbook of latent semantic analysis. Psychology Press.

[3] Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. the Journal of machine Learning research 3:993–1022.

[4] Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. Communications of the ACM, 18(11), 613–620.

[5] Gabrilovich, E., and Markovitch, S. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In IJCAI, volume 7, 1606–1611.

[6] https://github.com/StevenLOL/ialp2016_Shared_Task

[6] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26:3111–3119.

[7] Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huanbo Luan. 2015b. Joint learning of character and word embeddings. In *International Conference on Artificial Intelligence*.

[8] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information.

[9] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

[10] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2010. Deep sparse rectifier neural networks. *Journal of Machine Learning Research*, 15.

[11] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Computer Science*.

[12] Y. Freund. 1996. Experiments with a new boosting algorithm. In *Thirteenth International Conference on Machine Learning*, pages 148–156.

[13] Harris Drucker. 1997. Improving regressors using boosting techniques.

[14] Rong En Fan, Kai Wei Chang, Cho Jui Hsieh, Xiang Rui Wang, and Chih Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9(9):1871–1874.

[15] Chia Hua Ho and Chih Jen Lin. 2012. Large-scale linear support vector regression. *Journal of Machine Learning Research*, 13(1):3323–3348.

[16] Jerome H. Friedman. 2000. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232.

[17] Jerome H. Friedman. 2002. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378.

[18] Tianqi Chen, Tong He, and Michael Benesty. 2015a. xgboost: Extreme gradient boosting.

[19] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. *arXiv:1603.02754*.

[20] Eric Holloway, II Marks, et al. 2016. High dimensional human guided machine learning. *arXiv:1609.00904*.

[21] I. T. Jolliffe. 1986. Principal component analysis. *Springer Berlin*, 87(100):41–64.