TWINA at SemEval-2017 Task 4: Twitter Sentiment Analysis with Ensemble Gradient Boost Tree Classifier

Naveen Kumar Laskari
Assistant Professor of IT
BVRIT Hyderabad
naveen.laskari@gmail.com

Suresh Kumar Sanampudi
Head of the Department, IT
JNTUH College of Engineering Jagitial
sureshsanampudi@gmail.com

Abstract

This paper describes the TWINA system, with which we participated in SemEval-2017 Task 4B (Topic Based Message Polarity Classification – Two point scale) and 4D (Two-point scale Tweet Quantification). We implemented ensemble based Gradient Boost Trees Classification method for both the tasks. Our system could perform well for the task 4D and ranked 13th among 15 teams, for the task 4B our model ranked 23rd position.

1 Introduction

Twitter, as a social networking service and microblogging service has gained great success in the recent years. It attracted millions of users to disseminate most up-to-date information, which resulted in generating massive amounts of information every day. Users share their opinions and experience on Twitter with the limit of 140 characters length text called as Tweet. Many applications in the field of Natural Language processing (NLP) and Information Retrieval (IR) are suffering severely from noisy in such a short 140 character length text.

This paper describes the system, with which we participated in Task 4 (Sentiment Analysis in Twitter) of SemEval – 2017 (Rosenthal et al., 2017). Organizers have given five different subtasks in task 4, they are:

* Task-4A: Message Polarity Classification
* Task-4B: Two-point scale Topic Based Message Polarity Classification
* Task-4C: Five-point scale Topic-Based Message Polarity Classification
* Task-4D: Two-point scale-Tweet quantification
* Task-4E: Five-point scale - Tweet quantification

We participated in only two subtasks B and D. With our submissions, we could stand in 13th position among 15 participants of task 4D and ranked 23rd position in task 4B. For both the tasks B and D, we implemented basic model of ensemble based Gradient Boost Tree Classifier and applied parameter optimization technique to improve the results.

The rest of the paper is organized as follows: In section 2 we describe the datasets, section 3 pre-processing of data for analysis, section 4 describes the model implementation using ensemble based Gradient Boost Trees Classification technique, section 5 gives results and section 6 gives conclusion and future work.

2 Datasets

In implementing the solution for SemEval Task 4, for every subtask the organizers provide training, development testing and testing datasets for training and testing. In addition to, the organizers made 2015 datasets available for training and tuning. We have used 4896 tweets for training the model and 20632 to test the model during the development. Final test of the model has been done on 12284 tweets.

Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017), pages 659–663,
Vancouver, Canada, August 3 - 4, 2017.

3 Pre-processing

Twitter has a constraint that, Tweet should not exceed 140 characters to convey the information or message. This makes the users to use unpredictable ways of expressing themselves. To find out sentiment from these kinds of tweets is very challenging task. In addition to, short text users are using different emoticons to express their opinions and feelings. Dealing with emoticons is a challenging task. To get the better results, we have to apply some pre-processing steps in order to clean Tweets for not to have unnecessary information. Initially each tweet converted into lower case and all URLS and HTML parts, Hash tags are removed from these tweets. Basically, emoticons has considered as two categories SAD and HAPPY, to deal with emoticons, each of the emoticons has been replaced with its category label either SAD or HAPPY. The Table 1 shows how the pre-processing step is applied, for the original Tweet and pre-processed Tweet can be seen.

Original Tweet	Look @Qualcomm I found the 1st #Snapdragon Phone in my stuff from #Toshiba and @Microsoft. Still Working :) http://t.co/dLbuag6QDU
After Preprocessing	look found 1st snapdragon phone stuff toshiba still working HAPPY
Original Tweet	@darebeark @alyaeldeeb12345 my memory doesnt have more space. So i cant download it :(but i'll try to download it tomorrow from iPad
After Preprocessing	memory does not space cant download SAD try download tomorrow ipad
Original Tweet	Don't forget to collect the bills and win free ipod nano 7th generation & 17% CK vouchers of total bills , we...http://t.co/CNn4Ln9swy
After Preprocessing	do not forget collect bills win free ipod nano 7th generation 17% ck vouchers total bills

Table 1: Tweet Pre-processing

4 Implementation

To train and test our model implementations, we have downloaded the training, development testing and testing datasets provided by the SemEval-2017 Task 4 organizers. After pre-processing the Tweet, we extracted word2vec features using genism models. These word2vec features are used to train the Gradient Boost Tree Classifier (GBC). After training the GBC model, development test dataset has been used to validate the model and final test dataset has been used to evaluate the model.

4.1 Word2Vec

Word2vec[1] model is used for learning the vector representations of words called word embeddings (Mikolov et al., 2013; Pennington et al.,2014). Word2vec is computationally efficient predictive model for learning word embeddings. It comes in two flavors, the Continuous Bag-of-Words model (CBOW) and the Skip-Gram model. Algorithmically, these models are similar, except that CBOW predicts target words from source context words, while the skip-gram does the inverse and predicts source context-words from the target words. The amazing property of these word embeddings is that, it effectively captures the semantic meanings of the words.

4.2 Gradient Boost Tree Classifier

Gradient Boosting is a machine learning technique for regression and classification problems, it builds an ensemble of trees one-by-one, and then the predictions of individual trees are summed.
Gradient Boosting involves three elements:

- A loss function to be optimized
- A weak learner to make predictions
- An additive model to add weak learner to minimize the loss function.

Decision trees are used as weak learners in gradient boosting. Trees are constructed in greedy manner by choosing the best split points. Trees are added one at a time, and existing trees in the model are not changed.

As we have used Scikit-learn[2] for our model implementation. It is a free software library for machine learning in python. Scikit-learn come with various classification, regression and clustering techniques. It is designed to interoperate with Python numerical and scientific libraries NumPy and SciPy.

Gradient Boosting is typically used with decision trees. In constructing the decision trees in Gradient Boosting method various parameters are used for defining a tree are - 'n_estimators', 'max_depth', 'subsample', 'min_samples_leaf', 'learning_rate', 'random_state '.

min_samples_leaf is the minimum observations or samples required in leaf or terminal node. Lower values can be picked to control the over fitting problem and solve class imbalance problem, so we fixed with 1.

n_estimators is the number of sequential trees to be modeled. In GBC is fairly robust for the higher values of trees, but it can still over fit from point on. Hence, we checked various combinations of values and fixed with 2500.

max_depth is the maximum depth of the tree. Appropriate value has to be picked to control overfitting, because as the higher depth tree will allow the model to learn very specific relations, which leads to overfitting. So we fixed with 7.

subsample is the fraction of observations to be used for each construction. Selection of the subsample is done by purely random sampling approach. The value slightly less than 1 makes the model robust. We fixed at 0.75.

random_state is the random number seed used to generate the same random numbers every time. This is very important parameter. If we don't fix the random number, then we will have different outcomes for subsequent runs on the same parameters. We fixed with 3.

learning_rate is the parameter which determines the impact of each tree on the final outcome. Learning rate controls the magnitude of change in the estimates. Lower values are suitable to make the model more robust, but need to construct more number of trees to model all the relations, which actually computationally expensive. We fixed with 0.005.

We have tested Gradient Boost Tree Classifier model with various combinations of values for the above parameters, and for every combination the accuracy of the model has been evaluated. We could arrive at comparatively best results for the above combinations.

5 Results

We participated in only two sub tasks (Task 4B & 4D) of SemEval-2017 Task 4. We have used ensemble based Gradient Boost Trees Classification technique for both the subtasks. For Task 4B we classified the polarity of the Tweet with respect to a particular entity either positive or negative.

For Task 4D, we assigned the probability score for each Tweet and computed mean value of the positive and negative probabilities for entity level. The computed mean probability of the entity is considered as the final score for the Tweet quantification towards the entity.

	Precision	Recall	F1-Score
Positive	0.389	0.834	0.530
Negative	0.546	0.133	0.214
Average		0.483	0.372
Overall Score : 0.483			
Accuracy : 0.412			

Table 2: Results for subtask- 4B

[1]https://radimrehurek.com/gensim/models/word2vec.html

[2]http://scikitlearn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

[3]https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/

The organizers have defined various baselines for measuring the performance of submissions. For task 4B average recall and accuracy for each class is considered as baseline. For task 4D five baselines have been defined. Baseline 2 is macro-averaged KLD, AE and RAE on train, dev, devtest and test from 2016. Baseline 3 is micro-averaged KLD, AE and RAE on train, dev, devtest, and test from 2016. Baseline 4 is micro-averaged KLD, AE and RAE on train, dev, devtest and test from 2016. Baseline 5 is micro-averaged KLD, AE and RAE on train, dev, devtest and test from 2015 and 2016.

S.No	System	ρ	F_1^{PN}	Acc
1	BB_twtr	0.882	0.890	0.897
2	DataStories	0.856	0.861	0.869
3	Tweester	0.854	0.856	0.863
4	TopicThunder	0.846	0.847	0.854
5	TakeLab	0.845	0.836	0.840
6	funSentiment	0.834	0.824	0.827
7	YNU-HPCC	0.834	0.816	0.818
8	WarwickDCS	0.829	0.834	0.843
9	CrystalNest	0.827	0.822	0.827
10	Zhangweuda2080	0.826	0.830	0.838
11	Amobee-C-137	0.822	0.801	0.802
12	SINAI	0.818	0.806	0.809
13	NRU-HSC	0.798	0.787	0.790
14	EICA	0.790	0.775	0.777
15	OMAM	0.779	0.762	0.764
16	NileTMRG	0.769	0.774	0.789
17	EliRF-UPV	0.766	0.773	0.790
18	DUTH	0.663	0.600	0.607
19	Ej-za-2017	0.594	0.486	0.518
20	SSN_MLRGI	0.586	0.494	0.518
21	YNU_1510	0.516	0.499	0.499
22	TM_Gist	0.499	0.428	0.444
23	**SSK_JNTUH**	**0.483**	**0.372**	**0.412**
	Baseline 1 : All POSITIVE	**0.500**	**0.285**	**0.398**
	Baseline 2: All NEGATIVE	**0.500**	**0.376**	**0.602**

Table 3: Comparative Results for subtask- 4B

6 Conclusions and Future work

In this paper we presented TWINA system, with which we participated in two sub tasks of SemEval-2017. This is the first time we participated in SemEval Task; there is much scope for the improvement. We have used very simple feature extraction technique like word2vec, and ensemble based Gradient Boost Tree Classification method. We can get better results with the implementation of good feature engineering techniques and use of deep neural networks for classification task.

S. No	System	KLD	AE	RAE
1	BB_twtr	0.036	0.080	0.598
2	DataStories	0.048	0.095	0.848
3	TakeLab	0.050	0.096	1.057
4	CrystalNest	0.056	0.104	1.202
5	Tweester	0.057	0.103	1.051
6	funSentiment	0.060	0.109	0.939
7	NileTMRG	0.077	0.120	1.228
8	NRU-HSC	0.078	0.132	1.528
9	Ecnucsy	0.092	0.143	1.922
10	THU_HCSI_IDU	0.129	0.179	2.428
11	Amobee-C-137	0.149	0.179	2.168
12	OMAM	0.164	0.204	2.790
13	**SSK_JNTUH**	**0.421**	**0.314**	**2.983**
14	EliRF-UPV	1.060	0.593	7.991
15	YNU-HPCC	1.142	0.592	7.859
	Baseline 1	**1.518**	**0.422**	**2.645**
	Baseline 2	**0.554**	**0.423**	**6.061**
	Baseline 3	**0.591**	**0.432**	**6.169**
	Baseline 4	**0.534**	**0.418**	**6.000**
	Baseline 5	**0.587**	**0.431**	**6.157**

Table 4: Comparative Results for subtask- 4D

7 References

Chikersal, Prerna, Soujanya Poria, and Erik Cambria. "SeNTU: sentiment analysis of tweets by combining a rule-based classifier with supervised learning." Proceedings of the International Workshop on Semantic Evaluation, SemEval. 2015.

Kharde, Vishal, and Prof Sonawane. "Sentiment analysis of twitter data: A survey of techniques." arXiv preprint arXiv: 1601. 06971(2016).

Liu, Bing. "Sentiment analysis and opinion mining." Synthesis lectures on human language technologies 5.1 (2012): 1-167.

Meyer, David. "How exactly does word2vec work?." (2016).

Nakov, Preslav, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. "SemEval-2016 task 4: Sentiment analysis in Twitter." Proceedings of SemEval (2016): 1-18.

Roe, Byron P., et al. "Boosted decision trees as an alternative to artificial neural networks for particle identification." Nuclear Instruments and Method in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 543.2 (2005): 577-584.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 task 4: Sentiment analysis in Twitter. In Proceedings of the 11th International Workshop on Semantic Evaluation. Vancouver, Canada, SemEval ’17

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. SemEval-2015 task 10: Sentiment analysis in Twitter. In Proceedings of the 9th International Workshop on Semantic Evaluation. Denver, Colorado, SemEval ’15, pages 451–463.