PRHLT: Combination of Deep Autoencoders with Classification and Regression Techniques for SemEval-2015 Task 11

Parth Gupta
PRHLT Research Center
Universitat Politècnica de València
Camino de Vera, s/n
46022 Valencia, SPAIN
pgupta@dsic.upv.es

Jon Ander Gómez
PRHLT Research Center
Universitat Politècnica de València
Camino de Vera, s/n
46022 Valencia, SPAIN
jon@dsic.upv.es

Abstract

This paper presents the system we developed for Task 11 of SemEval 2015. Our system had two stages: The first one was based on deep autoencoders for extracting features to compactly represent tweets. The next stage consisted of a classifier or a regression function for estimating the polarity value assigned to a given tweet. We tested several techniques in order to choose the ones with the highest accuracy. Finally, three regression techniques revealed as the best ones for assigning the polarity value to tweets. We presented six runs corresponding to three regression different techniques in combination with two variants of the autoencoder, one with input as bags of words and another with input as bags of character 3-grams.

1 Introduction

Sentiment Analysis from texts is a growing field of research due to its social and economic relevance. Task 11 of SemEval-2015 (Semantic Evaluation Exercises) was proposed to the research community in order to foster the development of systems and techniques for Sentiment Analysis (Ghosh et al., 2015).

We faced this challenging task with a system based on deep autoencoders in combination with classification and regression techniques. We used deep autoencoders to extract features from tweets by means of two ways of splitting text: i) words and ii) character 3-grams. The training of autoencoders was unsupervised. The extracted features (10) and a few manually added features (5) were used for training classifiers or regression functions to estimate the tweet's polarity value.

The rest of the paper is organized as follows. Section 2 describes the proposed system. Section 3 presents the obtained results on the test set. Finally, conclusions are discussed in Section 4.

2 System Description

Our system consists of two stages: (1) Dimensionality reduction by means of deep autoencoders. (2) Polarity value assignment by using different classification and regression techniques.

The text of tweets was preprocessed before being used as input to the autoencoders. The autoencoders take as input a representation of each tweet. Two different representations were used: bags of words and bags of character 3-grams. In both cases the output of an autoencoder was a vector of 10 real values. Optionally we added other features in order to improve the polarity assignment, these additional features are binary features indicating whether some symbols or hash tags appear in the tweet. The idea behind adding these extra features is to set a context for learning under their influence. The different subsets of used features are described in subsection 2.3. The step of assigning a polarity value to a given tweet was carried out by a classifier or a regression function. Several techniques for classification and regression were tested. Table 2 shows the relation of the used techniques.

2.1 Tweets Preprocessing

As mentioned above, the input for the autoencoders was prepared from two different ways of splitting

Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), pages 689–693,
Denver, Colorado, June 4-5, 2015.

Pattern or regular expression	New text
"#"	" #"
"@"	" @"
"&"	" & "
"<"	"<"
">"	">"
"&.*;"	" HTML "
"\\u0092"	"'"
"[0-9]+:[0-9]+[Aa][Mm]"	" H "
"[0-9]+:[0-9]+"	" H "
"[0-9]+[Aa][Mm]"	" H "
"http[s]*://[a-zA-Z0-9\\./-_]+"	""
"http[s]*:/+"	""
"http"	""
"@[_a-zA-Z0-9]+"	""
" : "	" "
" [0-9\\.-:]+ "	" N "
"[\\u00ff-\\uffff]+"	" A "
"!!!+"	"!3+"
"\\?\\?\\?+"	"?3+"
"\\.\\.\\.+"	"?.+"
"\\p{Punct}{3,}"	" P "
"> >"	">"
"< <"	"<"
">+"	">"
"<+"	">"
"-+"	"-"
" +"	" "
" "	"_"
"_+"	"_"

Table 1: Substitution rules used for normalizing the text of tweets. The double quotes are used here as delimiters, like in Java for String literals, they are not part of the pattern. Rules are presented in the same format they were used as arguments for the method `replaceAll()` of class `String` of Java. Rules were applied in the same order they appear in this table.

the text of tweets. Before the splitting step, a cleaning process was carried out by applying a set of substitutions. The goal was to normalize the text before generating the bags of words or character 3-grams.

Table 1 shows the rules used for carrying out such substitutions. These rules were extracted by us after analyzing the text of tweets corresponding to the training set. The order in that these rules were applied was relevant to the final result. The desired effects were the following:

- Removing URLs from the text of tweets. We assumed URLs were not relevant for guessing the polarity.
- User identifiers were also removed.
- Emoticons and possible animations were also reduced to a capital `A`. We were interested in knowing whether they appear or not.
- Sequences of repeated symbols or punctuation signs were reduced to one instance or a sequence to indicate the repetition.
- Numbers or dates were reduced to a capital letter indicating their appearance.
- Some symbols were forced to be preceded by a white space in order to facilitate the posterior splitting into words.
- Sequences of several white spaces were reduced to one white space and all white spaces were converted to underscores.

After the normalizing step, the splitting step was carried out in order to prepare the input for deep autoencoders. Two splitting ways were applied, one for separating words using white spaces (or underscores) and another one using character sequences of size 3 (character 3-grams).

In the case a tweet was represented as a bag of words, all the words found in the training set were used. A special entry for out-of-vocabulary words was introduced into the word table for generating the bags of words. We considered tokens as words those including only letters from the Latin alphabet. Numbers or other symbols were not included.

In the case of representing tweets as bags of character 3-grams, only those that appeared three or more times in the training set were used. The remaining ones were considered as out-of-vocabulary. A special entry for out-of-vocabulary 3-grams was introduced into the 3-grams table for generating the bags of 3-grams.

2.2 Deep Autoencoders

Autoencoders provide an unsupervised way to learn low-dimensional embeddings of the data. Such representation can be used for discriminative tasks. We used a deep autoencoder to extract such features. The fundamental block of our autoencoder was the restricted Boltzman machine (RBM). We used the contrastive-divergence algorithm for pretraining the autoencoder followed by the fine-tuning to minimize the reconstruction error shown in Eq. 1 (Hinton and Salakhutdinov, 2006).

$$J = \|X - X'\|^2 \tag{1}$$

where, X is the original vector and X' is its reconstruction.

The architecture[1] of the autoencoder was $|X|$-200-100-100-10 and the sigmoid function was used to add non-linearity to the hidden layers except for the final layer which was linear. We used replicated softmax to model count data in the visible layer of the autoencoder (Hinton and Salakhutdinov, 2009).

2.3 Classification and Regression Techniques

Different classification and regression techniques were tested in order to figure out which ones were the more appropriated for estimating the polarity of tweets. This checking process was carried out with the training set. We used the Scikit-Learn toolkit (Pedregosa et al., 2011) for all the tested techniques.

Given the output of each autoencoder we used three different sets of features:

1. Just the vectors of 10 real values obtained from autoencoders. 10 features.
2. Same 10 features as above plus five binary features indicating whether some hash tags or symbols were present in the tweet. The additional five binary features corresponded to the presence of three hash tags `#irony`, `#sarcasm`, `#not`, and whether the tweet contains quotes or emoticons. In total 15 features.
3. Same 15 features as for the second set plus additional binary features for indicating whether any of the hash tags found in the training set was present in the tweet. In total 3580 (10+5+3565) features.

Table 2 shows the list of all classification and regression techniques used in the second stage of our system. All techniques are used from the Scikit-Learn toolkit (Pedregosa et al., 2011). For training each classifier or regression function we used the same input data, i.e. the feature vectors representing each tweet from the training set and its polarity.

The output of classifiers is the integer value of the polarity, but in the case of regression functions the output value is truncated to the nearest integer.

[1]We did not notice any difference in performance empirically with other configurations with a general caution that much larger number of parameters model might lead to over-fitting.

The different classification and regression techniques used in the second stage were configured with the default values for their hyperparameters (Pedregosa et al., 2011). Some variations of hyperparameters were tested, but no further improvements were observed. Our purpose was to check which techniques were more suitable.

A more exhaustive search in order to find optimal combinations of hyperparameters for each technique would be an interesting extension of this work.

3 Results

Tables 2 and 3 show the results obtained with the test set. The whole training set was used for training all the tested techniques. It could be observed that the best results were obtained by the Ensemble of Extremely Randomized Trees (or Extra-Trees) used for regression (Geurts et al., 2006). Other ensemble techniques presented similar results. Focusing our attention on Table 3 and comparing with the results shown in Table 2, it could be observed as two variants of SVMs get results similar to the best ones, but no significant improvements were observed when using the set of 3580 features.

4 Conclusions

We developed a system for participating in Task 11 of SemEval-2015 which consisted of two stages. In the first stage stage we used deep autoencoders for obtaining a compact representation of tweets. We tried three sets of features that were used as input for different classification and regression techniques.

Results obtained in average from the 10-fold cross-validation we carried out with the training set revealed that the three most appropriated techniques were three ensembles: Extremely Randomized Trees, Random Forest and Bagging of Decision Trees. The regression setting of these techniques performed better than that of classification.

The fact that the techniques which obtained the best results are purely non-parametric and have no weights for approximating the output value, tell us that the obtained compact representation of tweets by means of deep autoencoders needs more analysis. An effort in exploring more configurations of autoencoders will help us to obtain better compact representations, which we plan to do in future. We

Classification or Regression Technique	Cosine Similarity			
	3-grams 10	**words 10**	**3-grams 15**	**words 15**
Automatic Relevance Determination Regressor	0.469	0.451	0.462	0.525
Bayesian Ridge Linear Regressor	0.552	0.562	0.609	0.618
Elastic Net Regressor	0.544	0.557	0.541	0.557
Ensemble AdaBoost Regressor Exponential	0.311	0.332	0.377	0.351
Ensemble AdaBoost Regressor Linear	0.540	0.556	0.554	0.587
Ensemble AdaBoost Regressor Squared	0.199	0.213	0.314	0.212
Ensemble Bagging Regressor with Decision Trees	0.558	0.549	0.593	0.587
Ensemble of Extra Trees Classifier	0.549	0.542	0.549	0.537
Ensemble of Extra Trees Regressor	0.565	0.557	**0.623**	0.610
Ensemble of Random Forests Classifier	0.535	0.542	0.536	0.541
Ensemble of Random Forests Regressor	0.554	0.555	0.592	0.610
KNN Classifier with inverse distance weights	0.497	0.497	0.501	0.495
KNN Classifier with uniform weights	0.507	0.526	0.517	0.518
LARS Lasso Linear Regressor	0.546	0.546	0.546	0.546
Lasso Linear Regressor	0.545	0.557	0.545	0.557
Logistic Regression (Classifier)	0.556	0.542	0.545	0.541
Perceptron Classifier	0.469	0.451	0.462	0.525
Passive Aggresive Regressor	0.561	0.378	0.564	0.384
RANSAC Regressor	0.507	0.532	0.547	0.592
Ridge Linear Regressor	0.552	0.563	0.608	0.620
SVM Linear Classifier	0.555	0.539	0.552	0.545
SVM Linear Regressor	0.551	0.552	0.583	0.570
SVM Polynomial Classifier	0.545	0.539	0.540	0.550
SVM Polynomial Regressor	0.587	0.560	0.599	0.610
SVM RBF Classifier	0.541	0.542	0.541	0.538
SVM RBF Regressor	0.593	0.562	0.604	0.560

Table 2: Results of all the tested techniques for the two kind of inputs used for the deep autoencoder: 3-grams and words, and for feature sets with 10 and 15 features.

Classification or Regression Technique	Cosine Similarity	
	3-grams 3580 features	**words 3580 features**
Bayesian Ridge Linear Regressor	0.605	0.621
Ensemble Bagging Regressor with Decision Trees	0.595	0.605
Ensemble of Extra Trees Regressor	**0.626**	0.596
Ensemble of Random Forests Regressor	0.593	0.616
Ridge Linear Regressor	0.596	0.615
SVM Linear Regressor	0.598	0.593
SVM RBF Regressor	0.610	0.566

Table 3: Results of some of the tested techniques for the two kind of inputs used for the deep autoencoder: 3-grams and words, and for the feature set with 3580 features.

also plan to use the tweet polarity information during the fine-tuning stage of training as an additional supervised component.

Acknowledgments

The research work of the first author is supported by the FPI grant from UPV.

References

Pierre Geurts, Damien Ernst and Louis Wehenkel. 2006. Extremely Randomized Trees. *Machine Learning* (ISSN 0885-6125) vol. 63, no. 1, pp. 2–42. Kluwer Academic Publishers, 2006

Aniruddha Ghosh, Guofu Li, Tony Veale, Paolo Rosso, Ekaterina Shutova, Antonio Reyes and John Barnden. 2015. SemEval-2015 Task 11: Sentiment Analysis of Figurative Language in Twitter. In Proc. *Int. Workshop on Semantic Evaluation (SemEval-2015)*, Co-located with NAACL and *SEM, Denver, Colorado, US, June 4-5, 2015

Geoffrey Hinton and Ruslan Salakhutdinov. 2006. Reducing the Dimensionality of Data with Neural Networks. *Science* vol. 313, no. 5786, pp. 504–507, 2006

Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot and Edouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python, In *Journal of Machine Learning Research*, JMLR vol. 12, pp. 2825–2830, 2011

Geoffrey Hinton and Ruslan Salakhutdinov. 2009. Replicated Softmax: an Undirected Topic Model. In *Advances in Neural Information Processing Systems*, pp. 1607–1614, 2009