UW-FinSent at SemEval-2017 Task 5: Sentiment Analysis on Financial News Headlines using Training Dataset Augmentation

Vineet John
University of Waterloo
vineet.john@uwaterloo.ca

Olga Vechtomova
University of Waterloo
ovechtomova@uwaterloo.ca

Abstract

This paper discusses the approach taken by the UWaterloo team to arrive at a solution for the Fine-Grained Sentiment Analysis problem posed by Task 5 of SemEval 2017. The paper describes the document vectorization and sentiment score prediction techniques used, as well as the design and implementation decisions taken while building the system for this task. The system uses text vectorization models, such as N-gram, TF-IDF and paragraph embeddings, coupled with regression model variants to predict the sentiment scores. Amongst the methods examined, unigrams and bigrams coupled with simple linear regression obtained the best baseline accuracy. The paper also explores data augmentation methods to supplement the training dataset. This system was designed for Subtask 2 (News Statements and Headlines).

1 Introduction

The goal of this SemEval task is to identify fine-grained levels of sentiment polarity in financial news headlines and microblog posts. Specifically, the task aims at identifying bullish (optimistic) sentiment, expressing the belief that the stock price will increase, and bearish (pessimistic) sentiment, expressing the belief that the stock price will decline. The expressed sentiment is quantified as floating point values in the range of -1 (very negative/bearish) to 1 (very positive/bullish), with 0 denoting neutral sentiment. (Cortis et al., 2017). This paper describes our system developed for subtask 2 (News Statements and Headlines).

While developing the system for this subtask, we systematically evaluated a number of alternative solutions for each step in the pipeline. Specifically, we investigated different document vectorization approaches, such as N-gram models, TF-IDF and paragraph vectors. A number of regression models were evaluated, namely, Simple Linear Regression, Support Vector Regression and XGBoost Linear Regression.

One of the challenges with performing sentiment analysis in the financial domain is scarcity of training data. We explored different approaches to augment the training data provided by the task organizers with training data from other sources in the financial domain, as well as using out-of-domain sentiment resources.

2 Approach

The overall approach to predicting the sentiment of the test dataset headlines is detailed below.

- **Pre-Processing & Cleaning**

  This step is needed to simplify and sanitize the input set of headlines. In the context of this task, since the headlines were short snippets ranging from 5 to 15 words in length, the only pre-processing done was replacing the name of the organization being spoken of in the headlines, with a generic organization name, to reduce the feature space.

- **Text Vectorization**

  The objective is to vectorize the textual content of the headlines into a numeric representation that a statistical learning model can then be trained on. N-gram models, TF-IDF and Paragraph Vector implementations were explored for this purpose. N-gram models generally performed the best on the trial dataset, followed by TF-IDF, and Paragraph Vectors. Of the different N-gram configurations experimented with, word N-grams that

Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017), pages 872–876,
Vancouver, Canada, August 3 - 4, 2017.

used a combination of unigrams and bigrams achieved the best baseline scores. These techniques are further discussed in Section 3.

- **Statistical Model Learning**

  The objective is to use the vector representations of the headlines as features and learn a model to predict the sentiment scores. Simple Linear Regression, Support Vector Regression and XGBoost Linear Regression were the learning methods that were used. The linear regression methods consistently outperformed Support Vector Regression and XGBoost regression in experiments on the training dataset. These techniques are discussed in Section 4.

3 Document Vectorization

Document vectorization is needed to convert the text content of the SemEval headlines into a numeric vector representation that can be utilized as features, which can then be used to train a machine learning model on. The methods for vectorization used are listed in the subsections below.

3.1 N-gram Model

For the purpose of this task, a vectorizer implementation using Scikit-Learn (Pedregosa et al., 2011) was used to obtain vector representations of the SemEval headlines, since they have been proven to be an effective representation of textual content for sentiment classification in general (Wang and Manning, 2012).

3.2 TF-IDF Model

The TF-IDF implementation in Scikit-Learn (Pedregosa et al., 2011) was used to obtain vector representations of the SemEval headlines.

3.3 Paragraph Vector Model

A Paragraph Vector representation model is comprised of an unsupervised learning algorithm that learns fixed-size vector representations for variable-length pieces of texts such as sentences and documents (Le and Mikolov, 2014). The vector representations are learned to predict the surrounding words in contexts sampled from the paragraph. In the context of the SemEval headlines, the vector representations were learned for the complete headline.

Two distinct implementations were explored while attempting to vectorize the headlines using the Paragraph Vector approach.

- Doc2Vec: A Python library implementation in Gensim[1].
- FastText: A standalone implementation in C++ (Bojanowski et al., 2016) (Joulin et al., 2016).

Doc2Vec was the final choice that was opted for due to the ease of integration into the existing system. The paragraph embeddings for Doc2Vec are trained using the SemEval training headlines corpus.

4 Regression Models

Three different regression implementations were used to train models to predict the sentiment scores of the headlines:

- **Simple Linear Regression**

  This is the standard version of linear regression that simply learns the weights for the feature vector that minimize the cost function, which is represented as a Euclidean loss function.

- **Support Vector Regression**

  The idea of SVR is based on the computation of a linear regression function in a high dimensional feature space where the input data is mapped using a non-linear function (Basak et al., 2007).

  Instead of minimizing the observed training error, Support Vector Regression (SVR) attempts to minimize the generalization error bound so as to achieve generalized performance.

- **XGBoost Regression**

  This is an ensemble method for regression that coalesces several 'weak' learners into a single 'strong' learner by iteratively minimizing the least squares error or Euclidean loss incurred by the cost function (Chen and Guestrin, 2016).

  The hyper-parameters applicable are the regularization parameter (λ) and the gradient descent step-size / learning rate (α).

[1] https://radimrehurek.com/gensim/models/doc2vec.html

Vectorization Method	Learning Model	R^2 Score	Cosine Similarity
Unigrams & Bigrams	Simple Linear Regression	0.38	0.63
Unigrams & Bigrams	Support Vector Regression	0.38	0.63
Unigrams & Bigrams	XGBoost Regression	0.21	0.50
TF-IDF	Simple Linear Regression	-0.10	0.50
TF-IDF	Support Vector Regression	0.38	0.63
TF-IDF	XGBoost Regression	0.19	0.47
Doc2Vec	Simple Linear Regression	-4.69	0.04
Doc2Vec	Support Vector Regression	-0.05	0.08
Doc2Vec	XGBoost Regression	-0.06	0.06

Table 1: Experimental Results

The implementation library utilized for the Simple Linear Regression and Support Vector Regression techniques is Scikit-Learn (Pedregosa et al., 2011), whereas the XGB Python library was used for the XGBoost regression implementation.

5 Training Dataset Augmentation

A few different datasets were used to train the models on, in an attempt to identify the best representative training set. The dataset augmentation strategies used are enumerated below.

- **Article Content Expansion**

  To increase the number of features to train on, it was decided to retrieve the full text content of the articles corresponding to the article headlines. This was achieved by creating an application to search for the article headlines that were part of the training set using an online search engine, and to retrieve the full-text of the article by scraping the content from the source websites.

  This application is implemented in Java and is open-source[2]. The implementation can be extended to augment any set of headlines with the corresponding article content.

  The assumption made here is that the sentiment expressed in the article headline sufficiently proxies the sentiment in the actual article content.

- **Amazon Product Reviews**

  This corpus is a set of Amazon product reviews[3], each consisting of the review text and a star rating on the scale of 1-5. To normalize the dataset, the rating scores 1 & 2 are assumed to be associated with negative reviews, 3 with neutral and 4 & 5 with positive reviews. This score range was then mapped to a -1 to 1 scale to match the sentiment scores of the training data. In total, $100,000$ documents from this dataset were used to augment the existing training dataset.

- **Financial Phrasebank**

  This dataset is specific to the financial domain and is manually annotated (Malo et al., 2014). It is comprised of a set of financial snippets from stock market related news that have been annotated with the classes positive, negative and neutral.

  To normalize the labels, neutral was assigned a sentiment score of 0 and experiments were run for $positive \in (1, 0.5)$ and $negative \in (-1, -0.5)$.

None of the above strategies proved to be a good augmentation of the existing data, since their addition to the training datasets did not show any improvements in the overall cross-validated accuracy score.

6 System Implementation

The entire system was coded in Python with the use of the Scikit-Learn (Pedregosa et al., 2011), XGB and Gensim libraries. This includes a framework for automated testing of accuracy scores to arrive at the best hyper-parameters to be used for unigram & bigram word count combinations, as well as Doc2Vec hyper-parameters.

The system implementation includes all the plugins pertaining to the different document vector-

[2]https://github.com/v1n337/news-article-extractor
[3]http://jmcauley.ucsd.edu/data/amazon/

Vectorization Method	Learning Model	Cosine Similarity
Unigrams & Bigrams	Simple Linear Regression	0.644
Unigrams & Bigrams	XGBoost Regression	0.547

Table 2: SemEval Task 5 Submissions

ization techniques and statistical learning techniques discussed in sections 3 and 4 respectively.

The code is open source[4] and is available to replicate the results published in this paper along with the instructions to operate the system.

7 Experimental Results

For arriving at the baseline scores, an exhaustive set of tests were conducted using each of the document vectorization techniques in combination with the regression techniques described in the previous sections.

Using the automated test-suite included as part of the system, it was concluded that the Doc2Vec model performed best when the number of dimensions (features) of text is around 832 and the learning algorithm completes 40 passes before settling on a vector representation. It was also concluded, that a combinations of unigrams & bigrams had the best baseline accuracy scores for the training datasets.

The measure of accuracy used was the R^2 score, also called the co-efficient of determination. The R^2 score can be computed using the below formula:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - f_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

where y is the gold set score vector and f is the predicted score vector, and N is the number of test samples.

The experimental results for the Training and the Trial datasets are shown in Table 1. The best baselines scores seem to favor the simplest vectorization model, i.e. unigrams & bigrams.

8 System Evaluation

For the two submissions permitted by SemEval, the methods used for the submissions made are described in Table 2.

The evaluation was done using the task evaluation metric, the $cosine\_score$ (Cortis et al., 2017).

$$cosine\_score = cosine\_weight * cosine(G, P)$$

where

$$cosine(G, P) = \frac{\sum_{i=1}^{N} G_i * P_i}{\sqrt{\sum_{i=1}^{N} {G_i}^2} * \sqrt{\sum_{i=1}^{N} {P_i}^2}}$$

and

$$cosine\_weight = \frac{|P|}{|G|}$$

and G, P are the gold set scores and the predicted scores respectively, for N test samples.

The simplest model implemented, using Unigrams & Bigrams, combined with Simple Linear Regression, was what yielded the best performance by the system, with a cosine similarity score of 0.644.

9 Conclusions and Future Work

This paper has described the UW-FinSent system developed by the UWaterloo team for Task 5, Subtask 2 during SemEval 2017.

The experimental results indicate that the usage of simpler techniques like N-gram text vectorization and linear regression to predict the continuous-valued scores achieve better results than bag-of-words or deep learning feature extraction techniques.

A recurring topic that needed to be addressed during the progress on this task was the fact there there were no reliable datasets that could accurately augment the training set. In the future, we plan to develop automatic methods for generating high quality, sentiment-annotated training datasets for the financial domain.

Acknowledgements

The authors are grateful to the organizers for their support for this task. The authors would also like to thank (Malo et al., 2014) for sharing the Financial Phrasebank Dataset for the purposes of our evaluation.

References

Debasish Basak, Srimanta Pal, and Dipak Chandra Patranabis. 2007. Support vector regression. *Neural Information Processing-Letters and Reviews* 11(10):203–224.

[4] https://github.com/v1n337/semeval2017-task5

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606* .

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pages 785–794.

Keith Cortis, André Freitas, Tobias Dauert, Manuela Huerlimann, Manel Zarrouk, Siegfried Handschuh, and Brian Davis. 2017. Semeval-2017 task 5: Fine-grained sentiment analysis on financial microblogs and news. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, Vancouver, Canada, pages 517–533. http://www.aclweb.org/anthology/S17-2089.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759* .

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*. volume 14, pages 1188–1196.

Pekka Malo, Ankur Sinha, Pekka Korhonen, Jyrki Wallenius, and Pyry Takala. 2014. Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the Association for Information Science and Technology* 65(4):782–796.

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.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*. Association for Computational Linguistics, pages 90–94.