

# UofL at SemEval-2016 Task 4: Multi Domain word2vec for Twitter Sentiment Classification

Omar Abdelwahab, Adel Elmaghraby

Computer Science and Engineering

University of Louisville, Louisville, KY

{omar.abdelwahab, adel}@louisville.edu

## Abstract

In this paper, we present a transfer learning system for twitter sentiment classification and compare its performance using different feature sets that include different word representation vectors. We utilized data from a different source domain to increase the performance of our system in the target domain. Our approach was based on training various word2vec models on data from the source and target domains combined, then using these models to calculate the average word vector of all the word vectors in a tweet observation, then input the average word vector as a feature to our classifiers for training. We further developed one doc2vec model that was trained on the positive, negative and neutral tweets in the target domain only. We then used these models in calculating the average word vector for every tweet in the training set as a preprocessing step. The final evaluation results show that our approach gave a prediction accuracy on the Twitter2016 test dataset that outperformed two teams that were among the top 10 in terms of AvgF1 scores.

## 1 Introduction

Twitter sentiment analysis deals with classifying the polarity of a tweet as positive or negative or neutral. We have participated in Semeval 2016 Twitter Sentiment Analysis subtask-A. Where we had to predict the polarity of a whole tweet rather than part of a tweet. We have started off with feature engineering and data preprocessing. Then we have divided our task into five classification tasks. Then we worked on creating our feature sets that we later used to select the best classifier/feature set combination for

each classification task as will be explained further in the paper.

In section 2 we explain our transfer learning approach, system components, data preprocessing, aggregation approach, word2vec training and doc2vec training. In section 3, we present our experiments. Then in section 4, we review and discuss our results. Finally in section 5, we make conclusions and outline some future work.

## 2 Transfer Learning Approach

Our transfer learning approach was based on using data from an additional source domain to supplement the provided target domain twitter data to train our word2vec models. The resultant models were later used in calculating the average word2vec vector of each tweet in our twitter training set as will be explained in section 2.4. We have also developed four binary classifiers and one arbiter three class label classifier. The four binary classifiers as well as the fifth arbiter classifier were a combination of logistic regression and SVM classifiers as we will show in the following sections.

### 2.1 System Components and Data Sets

In this section, we will specify the system components that were used in predicting the final results that we have submitted to SEMEVAL Task number 4. We have trained six word2vec models that will be discussed in more detail in section 2.4.

After training our word2vec models, we have selected five classifiers based on experimentation results using the preprocessed combined Rosenthal et al. (2015) and Nakov et al. (2016) semeval training twitter data set.

Two SVM classifiers, one for classifying positive and negative tweets was trained on positive and negative tweets. The second was for classifying between negative and non-negative tweets and it was trained on negative and non-negative tweets. In addition, we have trained three Logistic Regression classifiers, one for classifying between positive, neutral sentiments and it was trained on positive and neutral tweets.

While the second classifier was trained on negative, neutral tweets for classifying between negative and neutral polarities. The third logistic regression classifier was for classifying between positive, negative, neutral tweets and it was trained on positive, negative and neutral tweets. Different sets of features were used with each classifier as we will discuss in section 2.3.

As mentioned earlier, we have combined the 2015 Semeval twitter sentiment analysis subtask B's training set and the 2016 Semeval twitter sentiment analysis subtask A's training set into one training set of 11,798 tweets named `full_training_set`. We then combined the Guestrin et al. (2015) amazon baby toys product reviews data set (source domain) which were around 150,000 reviews with the `full_training_set` to build our word2vec models. On the other hand, we used the `full_training_set` to train our doc2vec model as will be explained in section 2.5. After training our word2vec and doc2vec models, the `full_training_set` was preprocessed for training our five classifiers mentioned above.

The validation set used for every classifier in the ensemble had the same characteristics of the training set. For example, since the positive/negative classifier was trained on positive and negative tweets, its validation set contained positive and negative tweets only.

Finally, the output of our five classifiers is aggregated to produce the final system output as discussed in section 2.3.

## 2.2 Preprocessing and Feature Engineering

Before we have trained any of our classifiers, we have started with preprocessing our training data. Keeping in mind that any preprocessing we performed on the training set was applied on the test set. We started with the following preprocessing steps:

- Removed character repetitions by removing the character repetitions that could distract our classifiers. For example a word like LOOOOOOOL is replaced with LOL.
- Replaced patterns by replacing words like 'won't' with 'will not', 'can't' with 'cannot'. 're' with 'are', etc.
- Converted tweets to lower case by converting upper case characters to lower case.
- Replaced website links with URL so links to websites that start with `www.*` or `http?:/*` were replaced with URL symbol.
- Converted `@username` to `AT_USER` by replacing `@username` instances found in tweets with `AT_USER` for our classifiers to easily identify that a user is being referenced.
- Removed additional white spaces in order to remove any noise that might affect our classifiers' performance.
- Replaced hash tags and removed stop words. As we replaced hash tags with the same word without the hash. For example, `#fun` is replaced with `fun`. As hash tags can give useful information. Also stop words are removed.
- Replaced Non-Alphabets and question words with space. Then we replaced numbers with `$NUM` symbol.
- Performed stemming and lemmatization using porter stemmer and WordNet lemmatizer class in the NLTK library.
- We used Bing Liu et al. (2004) positive-word and negative-word lexicons that contained a list of thousands of words associated with positive sentiment and negative sentiment. We replaced positive words with `$po` and negative words with `$ne`. We also used the NRC Unigram lexicon that helped us in replacing words that are more likely to give a positive sentiment with 'HAPO' and words that will more likely result in a negative sentiment with 'HANE'.

- The source domain data was preprocessed in the same way the target domain data (tweets) were preprocessed. However, the source domain data were only used in training the word2vec and doc2vec models.
- After finishing with the initial preprocessing steps discussed previously, the training set and the validation set were parsed into Graphlab Sframes for further manipulation.
- Then the Unigrams, bigrams and trigrams of every tweet in the training set were calculated and stored in additional columns in the same Sframe.
- Furthermore, the TFIDF of every tweet was calculated and stored in a new column.
- The six trained word2vec models that will be discussed in section 2.4 were each used in calculating the word vectors of each word in a tweet observation in the training set.
- Afterwards, the word vectors generated by each word2vec model were later averaged to get six averaged word vectors for every tweet in the training set. These average vectors were later saved in columns named w2v\_vectors\_pos\_neg, w2v\_vectors\_pos\_neutral, w2v\_vectors\_neutral\_neg, w2v\_vectors\_pos\_ornot, w2v\_vectors\_neg\_ornot, w2v\_vectors\_neutral\_ornot which were later used as features to our classifiers.

### 2.3 Features used and Aggregation

We have categorized our feature sets into four categories. There are different feature sets in each category. The first category is called feature category 1 that includes only base features tfidf, unigram, bigram, and trigrams. Then a second category containing the base as well as word2vec vectors and it's called feature category 2. Then a third feature category that contains base features in addition to doc2vec vectors only named feature category 3. Finally, a fourth category that contains base features, word2vec vectors, and doc2vec vectors is called feature category 4.

The features included in every feature set is presented in Appendix A while the feature sets in each category is shown in Appendix B.

When it came to aggregating the predictions from the four binary classifiers, we gave the highest priority to the neg\_ornot classifier. As it had a reasonably high accuracy in discriminating between negative and non-negative tweets on the validation set. Therefore if the neg\_ornot classifier classifies a tweet as negative, the final output of the system will be negative. However, if it classifies a tweet as non-negative, a majority vote is taken between the other three remaining binary classifiers. If two classifiers classify a tweet as positive, then the final system output will be positive, similarly if two classifiers classify a tweet as neutral, then the final system prediction is neutral, and likewise if two classifiers agree that a tweet is negative then the final sentiment output is negative. In the case when we get a tie from the other three classifiers and the neg\_ornot classifier classifies a tweet as non-negative, then the fifth arbiter three class label classifier is deployed to give the final classification output of the system.

### 2.4 Word2Vec

When Training our word2vec models for our final system, we used the continuous bag-of-words architecture. We combined positive amazon reviews and positive tweets into one file and named it pos\_text, then combined the negative tweets and negative amazon reviews into a second file named neg\_text, and similarly we combined the neutral amazon reviews with the neutral tweets in a third file named neutral\_txt. Our first word2vec\_pos\_neg model was trained on the pos\_text and neg\_text files.

While, the word2vec\_pos\_neutral model was trained on the pos\_text and neutral\_text. Then the word2vec\_neg\_neutral model was trained on the neg\_text and neutral\_text. Furthermore, the other three word2vec models were trained to be used to distinguish between negative and nonnegative tweets, positive and non-positive tweets, neutral and non-neutral tweets using pos\_text, neg\_text and neutral\_text files. The word2vec\_neg\_ornot model was trained on the neg\_text, and non\_neg\_text (pos\_text and neutral\_text combined). Also, word2vec\_pos\_ornot model was trained on the pos\_text and non\_pos\_text (neg\_text and neu-

Classification	Classifier	Feature Set	Category	Accuracy
Neg-Nonnegative	SVM	6	2	0.894
Neg-Neutral	LR	7	3	0.778
Pos-neg	SVM	8	4	0.851
pos-neg-neutral	LR	12	4	0.610
Pos-Neutral	LR	2	2	0.713

**Table 1:** Final classifier/feature set combinations selected.

tral\_text combined). Lastly, word2vec\_neutral\_or-not was trained on neutral\_text and non\_neutral\_text (pos\_text and neg\_text combined).

## 2.5 Paragraph Vector

We have trained a doc2vec model on all the positive, negative and neutral tweets in the training set to test whether paragraph vectors would improve fscores and prediction accuracies on the validation set. After building our doc2vec distributed memory model, we used it in inferring the tweet vector of every tweet in the training set and stored the result in the ‘vectors\_doc2vec\_tweetsonly\_dm’ column. Column vectors\_doc2vec\_tweetsonly\_dm is then added as a feature to our feature sets shown in Appendix A.

## 3 Experiments

We have carried out a number of experiments to help in selecting the feature sets to use for each classifier as well as which classifier type (SVM or Logistic Regression or boosted trees) to use for every binary classification and for the arbiter classifier. Appendix C contains a flowchart describing the system structure. The flowchart illustrates our system’s sentiment prediction process.

Table 2.0 shows the validation prediction accuracies for each binary classifier and on the arbiter classifier when varying the feature set. The feature set, classifier combination was selected based on the combination that yielded the best validation prediction accuracies. However, when evaluating our whole system or our final system output, we use fscore as a measure of system performance and not the test set prediction accuracy.

We have set class weights to auto in all classifiers we trained to help protecting against data imbalance which would lead to misleading results.

	Classifier	Feature Set	Accuracy
negative-notnegative	SVM	Feature Set 6	<b>0.89382</b>
	LR	Feature Set 9	0.87354
	SVM	Feature Set 1	0.87309
	LR	Feature Set 1	0.87156
	SVM	Feature Set 9	0.87022
	LR	Feature Set 6	0.86688
	LR	Feature Set 7	0.85667
	SVM	Feature Set 7	0.85503
neg-neutral	LR	Feature Set 7	<b>0.77817</b>
	LR	Feature Set 10	0.77432
	SVM	Feature Set 7	0.77113
	SVM	Feature Set 3	0.75102
	SVM	Feature Set 10	0.75097
	LR	Feature Set 1	0.74909
	SVM	Feature Set 1	0.74182
	LR	Feature Set 3	0.73878
pos-neg	SVM	Feature set 8	<b>0.85058</b>
	SVM	Feature Set 2	0.84890
	LR	Feature Set 2	0.82418
	LR	Feature set 8	0.82184
	LR	Feature Set 7	0.74869
	SVM	Feature Set 7	0.74607
	LR	Feature Set 1	0.73469
	SVM	Feature Set 1	0.72886
pos-neg-neutral	LR	Feature Set 12	<b>0.60966</b>
	LR	Feature Set 5	0.60686
	BoostedTrees	Feature Set 1	0.60613
	BoostedTrees	Feature Set 7	0.60232
	BoostedTrees	Feature Set 5	0.60081
	LR	Feature Set 7	0.57907
	BoostedTrees	Feature Set 12	0.56137
	LR	Feature Set 1	0.55799
pos-neutral	LR	Feature Set 4	<b>0.71325</b>
	SVM	Feature Set 4	0.70602
	SVM	Feature Set 1	0.68333
	LR	Feature Set 1	0.67619
	SVM	Feature Set 7	0.66508
	SVM	Feature Set 11	0.66180
	LR	Feature Set 7	0.66033
	LR	Feature Set 11	0.65693

**Table 2:** Validation Accuracy for different classifier/Feature Set combinations.

Starting System Fscore on the test set	0.26
Final system Fscore on the test set	0.51

**Table 3:** Fscore of the whole system when using the least performing classifier/feature set combinations and when using the best performing classifier/feature set combination.

## 4 Results and Analysis

Table 3 shows the least Fscore of the system we started with (features set 1 for all classifiers in the system) and the final system fscore before submission on our test set. We used fscore as the overall system performance measure and not the system accuracy. However, we have put the validation accuracy as the top classifier/feature set combination selection criteria for the individual classifiers in our system while setting class weights to auto in all of our classifiers in our system. It is clear from Table 1.0 that feature categories 2 and 4 were associated with the best performing classifiers. As mentioned earlier feature category 2 uses word vectors in addition to the base features in feature set 1. While feature category 4 uses word vectors and paragraph vectors with feature set 1. Which indicates that the addition of paragraph vectors with word2vec vectors gave best validation accuracies with the positive/negative and the pos/neg/neutral classifiers. However, it did not give the best validation accuracies with the positive/neutral and negative/nonnegative classifiers. As Feature category 2 that uses word vectors with feature set 1 gave the best validation accuracies with the positive/neutral and the negative/nonnegative classifiers. Finally, using only paragraph vectors with feature set 1 yielded the best validation accuracy with the negative/neutral classifier. Even though our doc2vec model was only trained on data from the target domain (tweets), it managed to give slightly better validation accuracy than when using word2vec vectors trained on the source (amazon reviews) and target (tweets) domains combined. Nonetheless, for positive/negative, positive/neutral, negative/nonnegative, positive/negative/neutral classification using word2vec vectors that were generated by our word2vec models trained on the source and target domains combined gave better validation set accuracies than when using only the doc2vec vectors generated by our doc2vec model that was trained on the

full\_training\_set (tweets) with feature set 1 only. Since Le et al. (2014) concluded that paragraph vectors are competitive with the state of the art word representation methods. We inferred based on our results that combining the source and target data to train our word2vec models would give better results than when training them only on the target data. Thus the external source data helped in building word2vec models that gave us more powerful features when compared to those generated by doc2vec (paragraph vector) models trained only on the target data (tweets in full\_training\_set).

## 5 Conclusion

Our approach resulted in higher prediction accuracies on the 2016 twitter test data set outperforming eight teams that had better AvgF1scores. Two of the eight teams were in the top 10 in terms of AvgF1scores. In the future, we will focus more on cross domain word representation as illustrated in Bollegala et al. (2015) for improving our transfer learning approach.

## References

- Bing Liu and Minqing Hu. 2004. *Mining and Summarizing Customer Reviews*. In Proceedings of KDD'04, August 22-25, 2004, Seattle, Washington, USA.
- Quoc Le and Tomas Mikolov. 2014. *Distributed Representations of Sentences and Documents*. Proceedings of the 31<sup>st</sup> International Conference on Machine Learning, Beijing, China.
- Danushka Bollegala, Takanori Maehara, and Ken-ichi Kawarabayashi. 2015. *Unsupervised Cross-Domain Word Representation Learning*. In Proceedings of the 53<sup>rd</sup> Annual Meeting of the Association for Computational Linguistics and the 7<sup>th</sup> International Joint Conference on Natural Language Processing, Beijing, China.
- Carlos Guestrin, and Emily Fox. 2015. *Machine Learning Foundations: A Case Study Approach*. By the University of Washington on Coursera, Seattle, WA.
- Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif M 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 (SemEval 2015)*. Association for Computational Linguistics, Denver, Colorado.
- Preslav Nakov, Alan Ritter, Sara Rosenthal, Veselin Stoyanov and Fabrizio Sebastiani. 2016. SemEval-

2016 Task 4: Sentiment Analysis in Twitter. *In Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*. Association for Computational Linguistics, San Diego, California.

## Appendix A. Feature Sets.

The following table shows the contents of each feature Set referenced above.

Set	Features included
1	['tfidf','1gram features','2gram features','3gram features']
2	['tfidf','1gram features','2gram features','3gram features','w2v_vectors_pos_neg','w2v_vectors_pos_neutral','w2v_vectors_pos_or-not','w2v_vectors_neg_or-not']
3	['tfidf','1gram features','2gram features','3gram features','w2v_vectors_neutral_neg','w2v_vectors_neg_or-not','w2v_vectors_neutral_or-not']
4	['tfidf','1gram features','2gram features','3gram features','w2v_vectors_pos_neutral','w2v_vectors_pos_or-not','w2v_vectors_neutral_or-not']
5	['tfidf','1gram features','2gram features','3gram features','w2v_vectors_neg_or-not','w2v_vectors_pos_neg','w2v_vectors_pos_neutral','w2v_vectors_neutral_neg','w2v_vectors_pos_or-not','w2v_vectors_neutral_or-not']
6	['tfidf','1gram features','2gram features','3gram features','w2v_vectors_neg_or-not']
7	['tfidf','1gram features','2gram features','3gram features','vectors_doc2vec_tweetonly_dm']

8	['tfidf','1gram features','2gram features','3gram features','w2v_vectors_pos_neg','w2v_vectors_pos_neutral','w2v_vectors_pos_or-not','w2v_vectors_neg_or-not','vectors_doc2vec_tweetonly_dm']
9	['tfidf','1gram features','2gram features','3gram features','w2v_vectors_neg_or-not','vectors_doc2vec_tweetonly_dm']
10	['tfidf','1gram features','2gram features','3gram features','w2v_vectors_neutral_neg','w2v_vectors_neg_or-not','w2v_vectors_neutral_or-not','vectors_doc2vec_tweetonly_dm']
11	['tfidf','1gram features','2gram features','3gram features','w2v_vectors_pos_neutral','w2v_vectors_pos_or-not','w2v_vectors_neutral_or-not','vectors_doc2vec_tweetonly_dm']
12	['tfidf','1gram features','2gram features','3gram features','w2v_vectors_neg_or-not','w2v_vectors_pos_neg','w2v_vectors_pos_neutral','w2v_vectors_neutral_neg','w2v_vectors_pos_or-not','w2v_vectors_neutral_or-not','vectors_doc2vec_tweetonly_dm']

## Appendix B. Feature Categories.

The following table shows mapping of feature sets to categories.

Feature Sets	Feature Category
1	1
2, 3,4,5,6	2
7	3
8, 9,10,11,12	4

Appendix C. System Flowchart.

