Ternary Twitter Sentiment Classification with Distant Supervision and Sentiment-Specific Word Embeddings

Mats Byrkjeland  Frederik Gørvell de Lichtenberg  Björn Gambäck
Department of Computer Science
Norwegian University of Science and Technology
NO—7491 Trondheim, Norway
{latsbyr,frederikgdl}@gmail.com, gamback@ntnu.no

Abstract

The paper proposes the Ternary Sentiment Embedding Model, a new model for creating sentiment embeddings based on the Hybrid Ranking Model of Tang et al. (2016), but trained on ternary-labeled data instead of binary-labeled, utilizing sentiment embeddings from datasets made with different distant supervision methods. The model is used as part of a complete Twitter Sentiment Analysis system and empirically compared to existing systems, showing that it outperforms Hybrid Ranking and that the quality of the distant-supervised dataset has a great impact on the quality of the produced sentiment embeddings.

1 Introduction

Bengio et al. (2003) introduced word embeddings as a technique for representing words as low-dimensional real-valued vectors capturing the words’ semantic and lexical properties, based on ideas dating back to the 1950s (Firth, 1957). Collobert and Weston (2008) showed the utility of using pre-trained word embeddings, and after the introduction of word2vec (Mikolov et al., 2013), which is much faster to train than its predecessors, word embeddings have become ubiquitous. This effectuated a dramatic shift in 2016 at the International Workshop on Semantic Evaluation (SemEval), with eight of the top-10 Twitter Sentiment Analysis systems using word embeddings.

Word embeddings learn the representation of a word by looking at its contexts (word neighbours in a text), making it difficult to discriminate between words with opposite sentiments that appear in similar contexts, such as “good” and “bad”. Hence, Tang et al. (2014) presented Sentiment-Specific Word Embeddings (or Sentiment Embeddings), a model employing both context and sentiment information in word embeddings.

Training sentiment embeddings requires large amounts of sentiment annotated data. Manual annotation is too expensive for this purpose, so fast, automatic annotation is used to set low-quality (weak) labels on large corpora; a procedure referred to as distant supervision. The traditional approach is to use occurrences of emoticons to guess binary sentiment (positive / negative). Motivated by the possible performance gains of focusing on the ternary task (where tweets can also be classified as neutral), this paper compares distant supervision methods on a large corpus of tweets that can be used to train sentiment embeddings. To this end, a new model architecture was developed with a new loss function trained on three-way classified distant supervised data. Various lexicon-based sentiment classifiers are compared, with their performance as distant supervision methods tested as part of a complete Twitter Sentiment Analysis system, evaluating both prediction quality and speed.

The paper is laid out as follows: Section 2 introduces related work on word and sentiment embeddings. Section 3 describes the proposed model for training ternary sentiment embeddings. Section 4 introduces a set of distant supervision methods and a comparison between them. Section 5 explores the optimal setup for the Ternary Sentiment Embedding Model through hyperparameter searches and dataset comparisons, while Section 6 compares the model against baselines and other methods to establish its performance. Section 7 concludes and suggests future improvements.

2 Related Work

Recent years have seen a vast number of Twitter Sentiment Analysis (TSA) systems, mainly because SemEval since 2013 has featured a TSA task, providing training data and a platform to compare different systems. This data will be uti-
lized here and the results below will be compared to those of SemEval in Section 6.4. First, however, the models most directly related to the present work will be introduced: the Collobert and Weston model (Collobert et al., 2011), three Sentiment Embeddings models by Tang et al. (2014), and their Hybrid Ranking Model (Tang et al., 2016). These can be viewed as sequential refinements of each other and as predecessors of the Ternary Sentiment Embedding Model described in Section 3.

The Collobert and Weston (C&W) Model: Collobert et al. (2011) proposed a task-general multilayer neural network language processing architecture. It starts with a Lookup Layer, which extracts features for each word, using a window approach to tag one word at a time based on its context. The input vector is then passed through one or several Linear Layers that extract features from a window of words, treated as a sequence with local and global structure (i.e., not as a bag of words). The following layers are standard network layers: a HardTanh Layer adds some non-linearity to the model (Collobert and Weston, 2008) and a final Linear Layer produces an output vector with dimension equal to the number of classes.

When learning word embeddings from context information, the output vector has size 1. For each context used to train the model, a corrupted context is created by replacing the focus word with a random word from the vocabulary. Both the correct and the corrupted context windows are passed through the model, with the training objective that the original context window should obtain a higher model score than the corrupted by a margin of 1. This can be formulated as a hinge loss function:

\[ \text{loss}_{cw}(t, t') = \max(0, 1 - f_{cw}(t) + f_{cw}(t')) \]

where \( f_{cw} \) is the model score, \( t \) and \( t' \) are the original and corrupted context windows, and \( f_{cw} \) the model score.

Sentiment Embeddings: To improve word embeddings for sentiment analysis, Tang et al. (2014) introduced Sentiment-Specific Word Embeddings (SSWE). They enhanced the C&W word embedding model by employing massive amounts of distant-supervised tweets, assigning positive labels to tweets containing positive emoticons and negative to those containing negative emoticons. Tang et al. used three strategies to incorporate sentiment information in embeddings: two basic models that only look at sentence sentiment polarity, and a Unified Model which adds word context and C&W’s corrupted context window training.

Basic Model 1 uses C&W’s window-based approach, but with the top linear layer’s output vector elements defining probabilities over labels. A softmax activation layer is added to predict positive n-grams as \([1, 0]\) and negative as \([0, 1]\). This constraint is relaxed in Basic Model 2, which removes the softmax layer to handle more fuzzy distributions and uses a ranking objective function:

\[ \text{loss}_r(t) = \max\{0, 1 - \delta_s(t)f_0^s(t) + \delta_s(t)f_1^s(t)\} \]

where \( f_0^s \) and \( f_1^s \) are the predicted positive and negative scores, while \( \delta_s(t) \) reflects the gold sentiment polarity of the context window \( t \), with

\[ \delta_s(t) = \{1 : f^s(t) = [1, 0]\} \land \{-1 : f^s(t) = [0, 1]\} \]

Unified Model uses corrupted context window training with two objectives: the original context should get a higher language model score and be more consistent with the gold polarity annotation than the corrupted one. The loss function combines word contexts and sentence polarity:

\[ \text{loss}_{us}(t, t') = \alpha \cdot \text{loss}_{cw}(t, t') + (1 - \alpha) \cdot \text{loss}_{us}(t, t') \]

where \( 0 \leq \alpha \leq 1 \) weights the parts, \( \text{loss}_{cw} \) is the C&W loss function, and with \( \delta_s(t) \) as above:

\[ \text{loss}_{us}(t, t') = \max\{0, 1 - \delta_s(t)f_1^s(t) + \delta_s(t)f_1^s(t')\} \]

Hybrid Ranking Model (Tang et al., 2016) splits the top linear layer of the Unified Model into a context-aware layer that calculates a context score \( f_{cw} \) and a sentiment-aware layer calculating a sentiment score \( f^s \) for the input context window. The objective function only compares the predicted positive and negative score for the correct context window when calculating the loss:

\[ \text{loss}_{hy} = \alpha \cdot \text{loss}_r + (1 - \alpha) \cdot \text{loss}_{cw} \]

3 Ternary Sentiment Embedding Model

A new neural network model for training word embeddings called the Ternary Sentiment Embedding Model is proposed. The model extends the Hybrid Ranking Model by Tang et al. (2016) for training Sentiment-Specific Word Embeddings by also looking at tweets labeled as “neutral”, and consists of three bottom (core) layers and two top layers that work in parallel, as shown in Figure 1.

Core Layers: The first layers identical to those of the C&W model. As with that model, the objective of the context part of the Ternary Sentiment Embedding Model is to assign a higher score to a correct context window than a corrupted window:

\[ \text{loss}_c(t, t') = \max(0, m - f^c(t) + f^c(t')) \] (1)
where \( m \) is the margin \( (m = 1 \Rightarrow loss_s = loss_{cw}) \), \( t \) and \( t' \) the correct resp. corrupted context windows, and \( f^c(\cdot) \) the context linear layer’s score.

**Ternary Sentiment Linear Layer:** A new top linear layer is introduced to calculate sentiment scores. It outputs a vector of size 3, representing positive, negative and neutral scores for a given context window. The objective is to give a higher score to the value corresponding to the context’s label than the other possible labels. A new margin hinge loss function is used to train the model:

\[
loss_s(t) = \max(0, m - f^c_s(t) + f^s_{i1}(t)) + \max(0, m - f^c_s(t) + f^s_{i2}(t)) \tag{2}
\]

where \( t \) is a context window, \( m \) the margin, \( f^c_s(\cdot) \) the sentiment score for the currently labelled sentiment of the input context, and \( f^s_{i1}(\cdot) \) and \( f^s_{i2}(\cdot) \) the sentiment scores for the other two classes.

The model’s total loss function is a weighted linear combination of the hinge losses for the Sentiment Linear and Context Linear layers:

\[
loss(t, t') = \alpha \cdot loss_s(t) + (1 - \alpha) \cdot loss_c(t, t') \tag{3}
\]

**Model Training:** As in the C&W model, parameters of the neural network are trained by taking the derivative of the loss through backpropagation. Stochastic Gradient Descent (SGD) is used to update the model parameters. This means that samples, in this case context windows created from tweets, are randomly drawn from the training corpus, and the parameters are updated for each sample passed through the model, according to:

\[
w_t = w_{t-1} - l_r \cdot g_t \tag{4}
\]

where \( w_t \) is the value of the parameter \( w \) at time \( t \), \( g_t \) its gradient at time \( t \), and \( l_r \) the learning rate.

The model parameters are initialised as in Tang et al. (2016). Lookup layer parameters are initialised with values from the uniform distribution \( U(-0.01, 0.01) \), while hidden layer parameters are initialised using fan-in (Collobert et al., 2011), i.e., the number of inputs used by a layer, \( i \). The technique draws the initial parameters from a centred distribution with variance \( \sqrt{\frac{2}{i}} \). Fan-in is also used for the learning rate, with \( l_r \) for the hidden layers in Eqn. 4 divided by the fan-in, \( i \).

### 4 Distant Supervision of Tweets

The idea of distant supervision is to automatically label data in order to be able to leverage large amounts of it. These data are called distant supervised or weakly annotated, as the quality is not great, but the quantity is. To train sentiment embeddings, large amounts of weakly annotated tweets are needed. This section describes the approach of extracting weak labels from a corpus of collected tweets (about 547 million), and explains each of the sentiment analysis methods that are compared for distant supervision use.

The outputs are ranked using SemEval’s measures \( F^P_N \) (the average of the \( F_1 \)-scores for positive and negative samples) and AvgRec (the average of the recall scores for the three classes). While \( F^P_N \) and AvgRec have been used in SemEval for both binary and ternary classification, it is debatable how representative they are for the latter. Hence, the Macro \( F_1 \) metric used by Tang et al. (2016) will also be calculated. It extends \( F^P_N \) by averaging the \( F_1 \)-scores of all three classes.

**Emoticons and Emojis:** Go et al. (2009) automatically classified tweet sentiment using distant supervision based on a few positive (‘;D’, ‘;>:D’, ‘:’) and negative (‘;:', ‘;:-(' , ‘:') emoticons, while removing tweets containing both a positive and negative emoticon. This method was reimplemented in Python and adapted to the ternary task by classifying tweets containing none of the emoticons as neutral. Further, since the sets of emoticons used by Go et al. are quite sparse compared to the vast amount of emojis and emoticons available today, extended sets (“Emojis+”) were also created, as shown in the Appendix.
AFINN, TextBlob and VADER: These methods respectively use the AFINN (Nielsen, 2011), TextBlob, and VADER (Hutto and Gilbert, 2014) libraries to count tweet sentiment scores. For AFINN, tweets with a 0 sentiment score were classified as neutral, while those with scores greater and lower than 0 were classified as positive and negative, respectively. For TextBlob, tweets with subjectivity score less than a threshold \( \theta_s \) were defined as neutral; a threshold \( \theta_p \) was set to classify tweets with polarity \( < -\theta_p \) as negative and those with polarity \( > \theta_p \) as positive. VADER returns a 3D vector where each element represents a score for each sentiment class. The vector is normalized so that \( \text{positiveScore} + \text{negativeScore} + \text{neutralScore} = 1 \). Setting a confidence threshold \( \theta_c > 0.5 \) certifies that the other scores are below 0.5. If no score is \( > \theta_c \), the tweet is skipped. VADER also gives a compound score, a single sentiment score from \(-1 \) to \(1 \) (most positive).

The methods’ hyperparameters were tuned through grid searches, testing each value in increasing steps of 0.1. VADER struggled to classify positive and negative tweets as the threshold increased, and performed best at \( \theta_c = 0.1 \). TextBlob performed best with a low subjectivity threshold, with \( \theta_s = 0.1 \) and \( \theta_p = 0.3 \) chosen for the final classifier, as these values gave the best Macro F\( _1 \).

Combo Average: An ensemble of the AFINN, TextBlob, and VADER classifiers, with scores normalised to be in the \([-1, 1]\) range. For AFINN, its score is divided by \(5 \cdot n \) (the number of words in the tweet), since \([5]\) is the highest score a word can get. For VADER, the compound score is used, while TextBlob’s score is already normalised. The scores are combined using a weighted average: \((a \cdot \text{afinn} + b \cdot \text{vader} + c \cdot \text{textblob})/(a + b + c)\). A threshold \( \theta \) is set so that tweets with score \( > \theta \) are classified positive, those with score \( < -\theta \) negative, and all others neutral. Running a grid search as above to select the method’s four parameters, the combination achieving the top \( F_{1}^{\text{POS}} \) score was \( \{a = 0.0, b = 0.4, c = 0.4, \theta = 0.2\} \) (this is called Combo A below), while the Macro F\( _1 \) winner was \( \{a = 0.3, b = 0.1, c = 0.1; \theta = 0.1\} \) (Combo B).

Lexicon Classifier (LC): A Python port of the Lexicon Classifier of Fredriksen et al. (2018), using their best performing lexicon and parameters.\(^3\)

Evaluation: All manually annotated SemEval datasets from 2013 to 2016 were downloaded. They contain IDs for 50,333 tweets, but 10,251 of those had been deleted, while duplicates were removed, leaving 39,731 tweets for later classifier training (the second column of Table 1). For the distant supervision, further filtering removed retweets (i.e., copies of original tweets; including retweets might lead to over-representation of certain phrases), tweets containing symbols (mostly weather data), tweets containing URLs, and tweets ending with numbers (often spam). Then 28,452 tweets remained for evaluating the distant supervision methods, distributed as in Table 1 (note that only 16% of the total tweets are negative).

Comparisons of the methods with tuned parameters on all SemEval datasets merged into one (the 2013-2016-all dataset of Table 1) are shown in Table 2. We see that the top Macro F\( _1 \) score is 0.570, which does not seem very impressive. However, to our knowledge no previous sentiment analysis

\(^{1}\text{github.com/fnielsen/afinn}\)
\(^{2}\text{github.com/sloria/textblob}\)
\(^{3}\text{github.com/draperunner/fjlc}\)
research has been evaluated against the complete set of SemEval datasets, making the results hard to compare to other work. Evaluating each dataset individually, a trend could be observed with decreasing scores for later data, with a top Macro F\textsubscript{1} score on the 2013-test set of 0.628 compared to 0.578 on the 2016-test set. This is consistent with Fredriksen et al. (2018) who noted significantly dropping scores for tests on 2016 data, attributing this to those sets having more noise and annotation errors than earlier datasets.

The runtimes (ms) in Table 2 were obtained on a computer with four AMD Opteron 6128 CPUs and 125 GB RAM running Ubuntu 16.04 (note that the given runtimes do not include saving to file). The emoticon-based methods are very fast (0.09 and 0.11 ms/tweet), but their scores are substantially worse than the others. The ensemble methods are slow (2.26 and 2.27 ms/tweet), since they have to calculate the score of each component method.

5 Optimising the Model

In order to find the best performing configuration of the Ternary Sentiment Embedding Model, the hyperparameters were tested one-by-one through a search of manually selected values.

More than 500 million tweets had been collected at the time of the start of the experiments, with URLs, mentions, reserved words and numbers removed. The tweets were lower-cased and elongated words reduced to contain a maximum of three repeating characters. Using the distant supervision methods described above, the collection was iterated through, and the resulting datasets saved. To create even datasets for each method and label, three sets of 1M tweets from each sentiment class were extracted from the total datasets for each method (except for the Emoticon method, which only annotated 151,538 tweets as negative, so its datasets were limited to 150k tweets for each label). The model hyperparameter searches were performed using the dataset created by the Lexicon Classifier, since it was the top performer in Table 2. The following paragraphs give the results for each of the hyperparameters.

Context Window Size: Testing with window sizes in the range \([1, 9]\) showed the best performing size to be 3 (Macro F\textsubscript{1} = 0.6325). Tweets are typically short texts with informal language. It is possible that larger context windows will lead to the model considering excerpts that are too long for tweet lingo, and would fit better for more formal texts. However, the differences in the results were too small to draw any conclusions.

Embedding Length is the dimension of each word embedding vector. The larger the dimension, the more fine-grained information the vectors can hold. \(\{50, 75, 100, 125, 150\}\) dimensions were tested, with 150 performing best (Macro F\textsubscript{1} = 0.6249), indicating that larger embeddings result in better scores for the model. This is no surprise, as word embeddings as GloVe and word2vec are commonly trained with dimensions of 200 or 300. However, a length of 100 was selected, since larger embeddings only gave minor improvements but severely increased processing time.

Hidden Layer Size: For the Ternary Sentiment Embedding Model, the hidden layers are the Linear Layer and the HardTanh Layer. Experiments show a minimal impact of varying the hidden layer size \(\{10, 20, 30, 50, 100\}\) neurons, having a range on the score values of only 0.0046. The best performance (Macro F\textsubscript{1} = 0.6201) was achieved with size 100. These results correspond well to the claim by Collobert et al. (2011) that the size of the hidden layer, given it is of sufficient size, has limited impact on the generalisation performance. However, the size of the hidden layers has a significant impact on training runtime, so since the difference in score values were small, a hidden layer size of 50 was used in the final model.

Alpha is the weighting between the sentiment loss and the context loss in the combined loss function (Eqn. 3) used when training the model. \(\alpha\)-values in the range \([0.1, 1.0]\) were explored. The best score (Macro F\textsubscript{1} = 0.6310) was achieved for \(\alpha = 0.2\). This indicates that the contexts of the words are more important than the sentiment of the tweets. However, leaving out sentiment information altogether (\(\alpha = 0\)) gave the by far worse score (0.5400). Interestingly, leaving out context information (\(\alpha = 1\)) did not perform as badly (0.6069).

Learning Rate states how fast the neural network parameters are updated during backpropagation. A small rate makes the network slowly converge towards a possible optimal score, while a large rate can make it overshoot the optimum. Testing on values from 0.001 to 1.1, the best learning rate was 0.01 (Macro F\textsubscript{1} = 0.6231), although the total range of the scores was only 0.0412.
Margin defines how the scores should be separated in the loss functions of Eqn. 1 and Eqn. 2. Larger margins lead to similar scores for each sentiment class giving a larger total loss, with the model parameters being updated by a larger value during backpropagation. Experimenting with margins in the range \([0.5, 10.0]\), a value of 2.0 obtained the best Macro \(F_1\) (0.6188). It is hard to predict the impact of higher margins, but since the loss is greater when sentiment scores are close, this appears to give a better separation of words from tweets belonging to each sentiment class.

Distant Supervision Method: Using the above-selected hyperparameter values, the Ternary Sentiment Embedding Model was trained on the 3M tweet datasets created by using each distant supervision method (450k tweets for the Emoticon method). Performance over 1–20 epochs is shown in Figure 2. A top Macro \(F_1\) score of 0.6440 was obtained for LC after 10 epochs, but the scores vary notably for each epoch. For a more robust comparison, the Macro \(F_1\) scores were averaged over epochs 10 to 20, with LC again performing best (0.6383), but followed closely by the ensemble methods (Combo B: 0.6361, Combo A: 0.6352), VADER (0.6339), and AFINN (0.6296).

6 Evaluating the Final System

To evaluate the performance of the Ternary Sentiment Embedding Model, it was compared to the Hybrid Ranking Model by Tang et al. (2016) using different distant supervision methods, as well as to a range of baselines, among them other popular word embeddings models. Finally, the performance of the total Twitter Sentiment Analysis system was evaluated against the state-of-the-art.

The Twitter Sentiment Analysis system comprises the Ternary Sentiment Embedding Model and a linear kernel Support Vector Machine (SVM). The \(C\) parameter of the SVM classifier was set through a coarse search on values ranges from 0.001 to 1000 with the word embeddings produced by the Ternary Sentiment Embedding Model using the LC distant supervision dataset and trained for 20 epochs, followed by two finer searches around the best value of 0.01, covering value ranges of \([0.001, 0.009]\) and \([0.01, 0.09]\), with a \(C\) value of 0.006 giving the best performing classifier. A small \(C\) means the classifier favours more misclassified samples over separating samples by a large margin, indicating that it is hard to avoid misclassifying some samples. However, the differences in scores were very low even for large variations of the parameter, meaning the samples to classify are not easily linearly separable.

### 6.1 Comparing Distant Supervision Methods

The Ternary Sentiment Embedding Model was trained for 20 epochs using the different distant supervision methods and the produced sentiment embeddings tested using the SVM classifier using 10-fold cross validation on the unfiltered 39,731 tweets from the 2013-2016-all dataset (i.e., all the combined 2013–2016 SemEval datasets), 15,713 of which were positive (39.5%), 5,945 negative (15.0%), and 18,073 neutral (45.5%). Table 3 shows different metrics for the tests, sorted by descending Macro \(F_1\) score.

The Combo methods perform best in this comparison. By averaging over three methods, they can overcome weaknesses of their components. While the combo methods were not the top performers in the comparison of the distant supervision methods in Table 2, the ability to balance other weak classifiers seems to be important when used as distant supervision method for the proposed model. The results show that the Ternary Sentiment Embedding Model performs best when
trained on data from a distant supervision method that is good at classifying all tweets into all three sentiment classes. The emoticon methods and TextBlob have weaknesses when classifying tweets into one or more of the classes, hence yielding the worst results for the total system.

6.2 Comparison to Baselines

In order to see how well the final TSA system performs, it was compared to some existing sentiment analysis methods. The systems were also tested using 10-fold cross-validation on the unfiltered 2013-2016-all dataset (39,731 tweets). Table 4 shows the results for each TSA system using the Macro $F_1$ metric.

‘Random Uniform’ and ‘Random Weighted’ are two simple baselines, respectively created by picking a random label from a uniform probability distribution and by picking a random label from the same distribution as in the training set. The distant supervision classifiers are as above, except that the Emoticons and Emoji+ methods add the variation that tweets containing both negative and positive emoticons are regarded as neutral.

The word embeddings for Ternary Sentiment Embedding Model, GloVe and word2vec were all trained on the same set of 3M tweets, with 1M from each of the sentiment classes, assigned by the Lexicon Classifier distant supervision method. The GloVe model (Pennington et al., 2014) was used to train word embeddings of dimension 100. The word2vec (Mikolov et al., 2013) embeddings were trained using both the Continuous Bag-of-Words (CBOW) and the Continuous Skip-gram model, also with 100 dimensions. Word embeddings were also produced using the Hybrid Ranking Model of Tang et al. (2016) trained on a set of 3M tweets classified with the LC method, but using only tweets labelled as positive or negative when training word embeddings, with 1.5M tweets of each class. All the word embeddings were fed to the SVM classifier specified above.

As the results in Table 4 shows, the best Macro $F_1$ scores were achieved by the word embedding systems. The word embeddings produced by the Ternary Sentiment Embedding Model gave slightly better results than the word embeddings produced by the Continuous Bag-of-Words model, however, the difference is small.

One of the strengths of the word2vec models is that they require much less training time than

| Model                                       | $F_1$ |
|---------------------------------------------|-------|
| Ternary Sentiment Embedding Model           | .6036 |
| word2vec (CBOW)                             | .6015 |
| Hybrid Ranking Model (w/LC)                 | .5919 |
| word2vec (Skip-gram)                        | .5886 |
| LC                                          | .5706 |
| GloVe                                       | .5662 |
| Combo B                                     | .5621 |
| Combo A                                     | .5579 |
| AFINN                                       | .5381 |
| VADER                                       | .5286 |
| TextBlob                                    | .3826 |
| Random Weighted                             | .3315 |
| Random Uniform                             | .3174 |
| Emoji+                                      | .2542 |
| Emoticon                                    | .2462 |

Table 4: The final Ternary Sentiment Embedding Model compared to baselines (Macro $F_1$-scores)

larger neural network models such as the Collobert and Weston model and the Ternary Sentiment Embedding Model. The word2vec models used approximately three minutes, while the Ternary Sentiment Embedding Model used 24 hours to train on 3M tweets. This advantage of the word2vec models means that they could be trained using a much larger dataset, which would likely yield an even better performance.

The word2vec models do not utilise sentiment information of the tweets, which is necessary to create sentiment embeddings with the Ternary Sentiment Embedding Model. This is another advantage of the word2vec models, as they have no need for a separate distant supervision method. The word2vec models are, however, slightly outperformed by the Ternary Sentiment Embedding Model in terms of the final score, and with further optimisation the difference could increase.

6.3 Comparison to Hybrid Ranking Model

In order to compare the distant supervision performance of the sentiment embeddings produced by the Ternary Sentiment Embedding Model and the Hybrid Ranking Model of Tang et al. (2016), both architectures were trained for 20 epochs on 3M tweets weakly annotated using the different distant supervision methods of Section 4. The Ternary Sentiment Embedding Model was trained on tweets labelled as positive, negative or neutral, with 1M of each, with the hyperparameters stated in Section 5. The Hybrid Ranking Model only utilises tweets labelled as positive or neutral, and was as a result trained on 1.5M tweets of each sentiment class, using the hyperparameters
Table 5: Ternary Sentiment Embedding Model vs. Tang et al.’s Hybrid Ranking (Macro F$_1$-scores)

| Dataset   | Ternary Embedding | Hybrid Ranking |
|-----------|-------------------|---------------|
| AFINN     | .602              | .578          |
| Combo A   | .608              | .587          |
| Combo B   | .609              | .592          |
| Emoticon  | .504              | .528          |
| Emoji+    | .548              | .536          |
| LC        | .604              | .592          |
| TextBlob  | .584              | .575          |
| VADER     | .596              | .596          |
| TSA system| .655              | .634          |

Table 6: Comparison to top results from different SemEvals (F$_{PN}$ scores). Subscripts denote the ranking the system would have achieved each year.

| Year | Top SemEval result | Ternary Embedding |
|------|--------------------|-------------------|
| 2013 | .6902              | .6178$_{10}$      |
| 2016 | .633               | .5805$_{12}$      |
| 2017 | .685               | .6291$_{10}$      |

Given by Tang et al. (2016). The produced sentiment embeddings were fed to the SVM classifier and tested using 10-fold cross-validation over the 2013-2016-all SemEval dataset.

The results in Table 5 show that the Ternary Sentiment Embedding Model outperforms the Hybrid Ranking Model using all but two of the eight tested distant supervision methods. The Hybrid Ranking Model only performs significantly better than the proposed model on the Emoticon dataset. The Hybrid Ranking Model is trained using only tweets labelled as positive or negative, while the Ternary Sentiment Embedding Model also utilises neutral tweets. The Emoticon method performs well for classifying tweets as positive or negative, but not for neutral, meaning that the quality of the positive and negative tweets is likely higher than for neutral tweets. This possibly explains why the Hybrid Ranking Model performs better when using this method.

When using the more sophisticated distant supervision methods, the Ternary Sentiment Embedding Model outperforms the Hybrid Ranking Model, with the exception of VADER where the scores are identical. This indicates that the proposed model is able to better take advantage of sentiment information from a larger set of tweets, increasing performance when used for the ternary sentiment classification task.

To compare the entire Twitter Sentiment Analysis system performance to that of Tang et al. (2016), the unfiltered datasets from SemEval 2013 were chosen for the classifier optimisation, with training on the 7,109 tweet 2013-train set (distributed 2,660-1,010-3,439 positive-negative-neutral) and testing on the 2013-dev set (1,228 tweets distributed 430-245-553), as this was the validation set of the 2013 workshop.

Tang et al. (2016) trained sentiment embeddings on 5M positive and 5M negative distant-supervised tweets, publishing the results produced by their model when tested with a SVM classifier on the SemEval 2013 test dataset, as presented in the last line of Table 5. The results indicate that the Ternary Sentiment Embedding Model performs better on the ternary classification task than the Hybrid Ranking Model, even though Tang et al.’s embeddings were trained on a much larger dataset than those used in the present work.

6.4 Comparison to SemEval

To see how the final Twitter Sentiment Analysis system fares against the state-of-the-art, its performance was compared to the published results of SemEval 2013 Task 2B (Nakov et al., 2013), SemEval 2016 Task 4A (Nakov et al., 2016), and SemEval 2017 Task 4A (Rosenthal et al., 2017).

The system was trained using the training sets provided by the respective workshop. For 2013, the model was trained on 2013-train-A and tested on 2013-test-A. SemEval 2016 and 2017 allowed training on the training and development datasets of previous years, so for 2016, the model was trained on a combined 2013-2016-train-dev-A dataset and tested on 2016-test-A, while for 2017, the model was trained on all 2013-2016 datasets and tested on 2017-test-A.

The results in Table 6 show that the Ternary Sentiment Embedding Model does not match the top systems of the different years. There are some possible reasons to this: The SemEval systems might have trained on other or more data than here. As tweets have been deleted, not as many could be downloaded as were available at the time of each workshop. Also, the model is optimised for Macro F$_1$ score. Had it been optimised for F$_{PN}$, better scores for this metric could have been obtained.

---

*Only the most similar systems are compared here; Tang et al.’s results improved by using additional lexical features.*
7 Conclusion and Future Work

The paper has proposed the Ternary Sentiment Embedding Model, a model for training sentiment-specific word embeddings using distance supervision. The model is based on the Hybrid Ranking Model of Tang et al. (2016), but considers the three classes positive, negative and neutral instead of just positive and negative. Experiments show the Ternary Sentiment Embedding Model to generally perform better than the Hybrid Ranking Model, and that the quality of the distance-supervised dataset greatly impacts the quality of the produced sentiment embeddings, and transi- 

tively the Twitter Sentiment Analysis system.⁶

The Hybrid Ranking Model only performed sig- 
nificantly better than the proposed model on the Emoticon dataset. Tang et al. (2016) use a dis- 

tant supervision method similar to the Emoticon method, due to the high precision that method can 
give. For a ternary model, however, it is not suffi- 
cient to only find some tweets that are likely pos- 
itive or negative, and a more sophisticated dis- 
tance supervision method is essential. This also 
means that a much larger and more varied corpus of distant supervised tweets can be used for train- 
ing, since no tweets are discarded. Consequently, 
the Ternary Sentiment Embedding Model outper- 
formed the Hybrid Ranking Model when using more sophisticated distant supervision methods.

Both Hybrid Ranking and Ternary Sentiment 

Embedding assume that all senses of a word are synonyms and that all words in a tweet have the same sentiment, ignoring their prior sentiment po- 

larity. Ren et al. (2016) proposed a model for training topic-enriched multi-prototype word em- 

beddings that addresses the issue of polysemy, 
significantly improving upon the results of Sem- 

eval 2013 on the binary classification task. Xiong 

(2016) addressed the prior polarity problem by ex- 

ploiting both a sentiment lexicon resource (Hu and 

Liu, 2004) and distant supervised information in 

a multi-level sentiment-enriched word embedding 

learning method. Further work could look at ex- 
tending the Ternary Sentiment Embedding Model 

with the ability to discriminate sentiment of pol- 
yseymous words in three classes, and to use word- 
sense aware lexica in order to combine the works of Ren et al. (2016) and Xiong (2016).

Acknowledgements

Thanks to Valerij Fredriksen and Brage Ekroll 

Jahren for providing their classifier code, to the or- 

ganisers of the different SemEval sentiment analy- 
sis tasks for collecting the data, to the anonymous 

reviewers for comments that helped enrich the dis- 
cussion of the results, and to Steven Loria, Finn 

Arup Nielsen, Clayton J. Hutton and Eric Gilbert 

for respectively providing the TextBlob, AFINN 

and VADER libraries.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and 

Christian Jauvin. 2003. A neural probabilistic lan- 
guage model. Journal of Machine Learning Re- 

search, 3:1137–1155.

Ronan Collobert and Jason Weston. 2008. A unified 

architecture for natural language processing: Deep 

neural networks with multitask learning. In Pro- 
ceedings of the 25th International Conference on 

Machine Learning, pages 160–167, Helsinki, Fin- 

land. ACM.

Ronan Collobert, Jason Weston, Léon Bottou, Michael 

Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. 
Natural language processing (almost) from scratch. 
Journal of Machine Learning Research, 12:2493–2537.

John Rupert Firth. 1957. A synopsis of linguistic the- 
ory 1930–55. Studies in Linguistic Analysis, Special 
Volume of the Philological Society.

Valerij Fredriksen, Brage Jahren, and Björn Gambäck. 
2018. Utilizing large Twitter corpora to create sen- 
timent lexica. In Proceedings of the 11th Interna- 
tional Conference on Language Resources and Eval- 
uation, pages 2829–2836, Miyazaki, Japan. ELRA.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twit- 
ter sentiment classification using distant supervision. 
CS224N project report, Stanford University, CA, 
USA.

Minqing Hu and Bing Liu. 2004. Mining and summa- 
rizing customer reviews. In Proceedings of the Tenth 
ACM SIGKDD International Conference on Knowl- 
edge Discovery and Data Mining, pages 168–177, 
New York, NY, USA. ACM.

Clayton J. Hutto and Eric Gilbert. 2014. VADER: A 
 parsimonious rule-based model for sentiment analy- 
sis of social media text. In Proceedings of the Eighth International Conference on Weblogs 
and Social Media, pages 216–225, Ann Arbor, MI, USA. 
The AAAI Press.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey 
Dean. 2013. Efficient estimation of word representa- 
tions in vector space. CoRR, abs/1301.3781.

⁶To perform the experiments, several tools and programs 

were developed, most of these are open sourced. See: 
github.com/draperunner
Appendix:

Positive and negative emoticons and emojis

The character combinations and Unicode characters used in the ‘Emoticons’ and ‘Emojis’ distant supervision methods described in Section 4.

| Positive Emoticons | Emojis | Negative Emoticons | Emojis |
|-------------------|--------|-------------------|--------|
| :) :) | 😊😊 | :( DX | 😞😞 |
| :D :D | 😊😊 | :I :I | 😞😞 |
| =D :] | 😊😊 | :i :i | 😞😞 |
| [:3 | 😊😊 | :> :> | 😞😞 |
| :> 8-) | 😊😊 | :>/> :>/> | 😞😞 |
| 8) :} | 😊😊 | :e :\ | 😞😞 |
| :} :o | 😊😊 | :e =\ | 😞😞 |
| =] =] | 😊😊 | :< :\ | 😞😞 |
| :D 8-D | 😊😊 | :L =L | 😞😞 |
| 8D x-D | 😊😊 | :l :S | 😞😞 |
| xD X-D | 😊😊 | :</: | 😞😞 |
| XD =D | 😊😊 | =S :S | 😞😞 |
| 3B~D | 😊😊 | =\ =\ | 😞😞 |
| :(-) :') | 😊😊 | @ v.v | 😞😞 |
| :') :* | 😊😊 | >( D-: | 😞😞 |
| :* :x | 😊😊 | D:< | 😞😞 |
| :-) :) | 😊😊 | D: | 😞😞 |
| 3) 3) | 😊😊 | D8 | 😞😞 |
| 3) 3) | 😊😊 | D; | 😞😞 |
| ;D <3 | 😊😊 | D= | 😞😞 |