Amobee at SemEval-2017 Task 4: Deep Learning System for Sentiment Detection on Twitter

Alon Rozental[*] **, Daniel Fleischer**[*]
Amobee, Tel Aviv, Israel
{arozental,danielf}@amobee.com

Abstract

This paper describes the Amobee sentiment analysis system, adapted to compete in SemEval 2017 task 4. The system consists of two parts: a supervised training of RNN models based on a Twitter sentiment treebank, and the use of feedforward NN, Naive Bayes and logistic regression classifiers to produce predictions for the different sub-tasks. The algorithm reached the 3rd place on the 5-label classification task (sub-task C).

1 Introduction

Sentiment detection is the process of determining whether a text has a positive or negative attitude toward a given entity (topic) or in general. Detecting sentiment on Twitter—a social network where users interact via short 140-character messages, exchanging information and opinions—is becoming ubiquitous. Sentiment in Twitter messages (tweets) can capture the popularity level of political figures, ideas, brands, products and people. Tweets and other social media texts are challenging to analyze as they are inherently different; use of slang, mis-spelling, sarcasm, emojis and co-mentioning of other messages pose unique difficulties. Combined with the vast amount of Twitter data (mostly public), these make sentiment detection on Twitter a focal point for data science research.

SemEval is a yearly event in which teams compete in natural language processing tasks. Task 4 is concerned with sentiment analysis in Twitter; it contains five sub-tasks which include classification of tweets according to 2, 3 or 5 labels and quantification of sentiment distribution regarding topics mentioned in tweets; for a complete description of task 4 see Rosenthal et al. (2017).

This paper describes our system and participation in all sub-tasks of SemEval 2017 task 4. Our system consists of two parts: a recurrent neural network trained on a private Twitter dataset, followed by a task-specific combination of model stacking and logistic regression classifiers.

The paper is organized as follows: section 2 describes the training of RNN models, data being used and model selection; section 3 describes the extraction of semantic features; section 4 describes the task-specific workflows and scores. We review and summarize in section 5. Finally, section 6 describes our future plans, mainly the development of an LSTM algorithm.

2 RNN Models

The first part of our system consisted of training recursive-neural-tensor-network (RNTN) models (Socher et al., 2013).

2.1 Data

Our training data for this part was created by taking a random sample[1] from Twitter and having it manually annotated on a 5-label basis to produce fully sentiment-labeled parse-trees, much like the Stanford sentiment treebank. The sample contains twenty thousand tweets with sentiment distribution as following:

	v-neg.	neg.	neu.	pos.	v-pos.
Train	8.4%	23.2%	31.7%	25.3%	11.4%
Test	8.6%	23.0%	33.2%	24.8%	10.4%

2.2 Preprocessing

First we build a custom dictionary by means of crawling Wikipedia and extracting lists of brands,

---

[*]These authors contributed equally to this work.

[1]Amobee is an official Twitter partner and as such has access to its global stream of data.

celebrities, places and names. The lists were then pruned manually. Then we define the following steps when preprocessing tweets:

1. Standard tokenization of the sentences, using the Stanford coreNLP tools (Manning et al., 2014).

2. Word-replacement step using the Wiki dictionary with representative keywords.

3. Lemmatization, using coreNLP.

4. Emojis: removing duplicate emojis, clustering them according to sentiment and replacing them with representative keywords, e.g. "happy-emoji".

5. Regex: removing duplicate punctuation marks, replacing URLs with a keyword, removing Camel casing.

6. Parsing: parts-of-speech and constituency parsing using a shift-reduce parser[2], which was selected for its speed over accuracy.

7. NER: using entity recognition annotator[3], replacing numbers, dates and locations with representative keywords.

8. Wiki: second step of word-replacement using our custom wiki dictionary.

2.3 Training

We used the Stanford coreNLP sentiment annotator, introduced by Socher et al. (2013). Words are initialized either randomly as d dimensional vectors, or given externally as word vectors. We used four versions of the training data; with and without lemmatization and with and without pre-trained word representations[4] (Pennington et al., 2014).

2.4 Tweet Aggregation

Twitter messages can be comprised of several sentences, with different and sometimes contrary sentiments. However, the trained models predict sentiment on individual sentences. We aggregated the sentiment for each tweet by taking a linear combination of the individual sentences comprising the

tweet with weights having the following power dependency:

$$h(f, l, \text{pol}) = (1 + f)^{\alpha} \, l^{\beta} \, (1 + \text{pol})^{\gamma} + 1, \quad (1)$$

where α, β, γ are numerical factors to be found, f, l, pol are the fraction of known words, length of the sentence and polarity, respectively, with polarity defined by:

$$\text{pol} = |10 \cdot \text{vn} + \text{n} - \text{p} - 10 \cdot \text{vp}|, \quad (2)$$

where vn, n, p, vp are the probabilities as assigned by the RNTN for very-negative, negative, positive and very-positive label for each sentence. We then optimized the parameters α, β, γ with respect to the true labels.

2.5 Model Selection

After training dozens of models, we chose to combine only the best ones using stacking, namely combining the models output using a supervised learning algorithm. For this purpose, we used the Scikit-learn (Pedregosa et al., 2011) recursive feature elimination (RFE) algorithm to find both the optimal number and the actual models, thus choosing the best <u>five</u> models. The models chosen include a representative from each type of the data we used and they were:

- Training data without lemmatization step, with randomly initialized word-vectors of size 27.

- Training data with lemmatization step, with pre-trained word-vectors of size 25.

- 3 sets of training data with lemmatization step, with randomly initialized word-vectors of sizes 24, 26.

The five models output is concatenated and used as input for the various tasks, as described in 4.1.

3 Features Extraction

In addition to the RNN trained models, our system includes feature extraction step; we defined a set of lexical and semantical features to be extracted from the original tweets:

- In-subject, In-object: whether the entity of interest is in the subject or object.

- Containing positive/negative adjectives that describe the entity of interest.

---

[2]http://nlp.stanford.edu/software/srparser.shtml.
[3]http://nlp.stanford.edu/software/CRF-NER.shtml.
[4]Twitter pre-trained word vectors were used, http://nlp.stanford.edu/projects/glove/

- Containing negation, quotations or perfect progressive forms.

For this purpose, we used the Stanford deterministic coreference resolution system (Lee et al., 2011; Recasens et al., 2013).

4 Experiments

The experiments were developed by using Scikit-learn machine learning library and Keras deep learning library with TensorFlow backend (Abadi et al., 2016). Results for all sub-tasks are summarized in table 1.

4.1 General Workflow

For each tweet, we first ran the RNN models and got a 5-category probability distribution from each of the trained models, thus a 25-dimensional vector. Then we extracted sentence features and concatenated them with the RNN vector. We then trained a Feedforward NN which outputs a 5-label probability distribution for each tweet. That was the starting point for each of the tasks; we refer to this process as the pipeline.

4.2 Task A

The goal of this task is to classify tweets sentiment into three classes (negative, neutral, positive) where the measured metric is a macro-averaged recall.

We used the SemEval 2017 task A data in the following way: using SemEval 2016 TEST as our TEST, partitioning the rest into TRAIN and DEV datasets. The test dataset went through the previously mentioned pipeline, getting a 5-label probability distribution.

We anticipated the sentiment distribution of the test data would be similar to the training data—as they may be drawn from the same distribution. Therefore we used re-sampling of the training dataset to obtain a skewed dataset such that a logistic regression would predict similar sentiment distributions for both the train and test datasets. Finally we trained a logistic regression on the new dataset and used it on the task A test set. We obtained a macro-averaged recall score of $\rho = 0.575$ and accuracy of $Acc = 0.587$.

Apparently, our assumption about distribution similarity was misguided as one can observe in the next table.

	Negative	Neutral	Positive
Train	15.5%	41.1%	43.4%
Test	32.3%	48.3%	19.3%

4.3 Tasks B, D

The goals of these tasks are to classify tweets sentiment regarding a given entity as either positive or negative (task B) and estimate sentiment distribution for each entity (task D). The measured metrics are macro-averaged recall and KLD, respectively.

We started with the training data passing our pipeline. We calculated the mean distribution for each entity on the training and testing datasets. We trained a logistic regression from a 5-label to a binary distribution and predicted a positive probability for each entity in the test set. This was used as a prior distribution for each entity, modeled as a Beta distribution. We then trained a logistic regression where the input is a concatenation of the 5-labels with the positive component of the probability distribution of the entity's sentiment and the output is a binary prediction for each tweet. Then we chose the label—using the mean positive probability as a threshold. These predictions are submitted as task B. We obtained a macro-averaged recall score of $\rho = 0.822$ and accuracy of $Acc = 0.802$.

Next, we took the predictions mean for each entity as the likelihood, modeled as a Binomial distribution, thus getting a Beta posterior distribution for each entity. These were submitted as task D. We obtained a score of $KLD = 0.149$.

4.4 Tasks C, E

The goals of these tasks are to classify tweets sentiment regarding a given entity into five classes—very negative, negative, neutral, positive, very positive—(task C) and estimate sentiment distribution over five classes for each entity (task E). The measured metrics are macro-averaged MAE and earth-movers-distance (EMD), respectively.

We first calculated the mean sentiment for each entity. We then used bootstrapping to generate a sample for each entity. Then we trained a logistic regression model which predicts a 5-label distribution for each entity. We modified the initial 5-label probability distribution for each tweet using

Task	A	B	C	D	E
	3-class.	2-class.	5-class.	2-quant.	5-quant.
Metric	ρ	ρ	MAE^M	KLD	EMD
Score	0.575	0.822	0.599	0.149	0.345
Rank	27/37	11/23	3/15	11/15	6/12

Table 1: Summary of evaluation results, metrics used and rank achieved, for all sub tasks. ρ is macro-averaged recall, MAE^M is macro-averaged mean absolute error, KLD is Kullback-Leibler divergence and EMD is earth-movers distance.

the following formula:

$$p^{\text{new}}(t_0, c_0) = \sum_{c \in C} \frac{p(t_0, c) \cdot p^{\text{entity-LR}}(t_0, c_0)}{\sum_{t \in T} p(t, c)}, \quad (3)$$

where t_0, c_0 are the current tweet and label, $p^{\text{entity-LR}}$ is the sentiment prediction of the logistic regression model for an entity, T is the set of all tweets and $C = \{\text{vn, n, neu, p, vp}\}$ is the set of labels. We trained a logistic regression on the new distribution and the predictions were submitted as task C. We obtained a macro-averaged MAE score of $MAE^M = 0.599$.

Next, we defined a loss function as follows:

$$\text{loss}(t_0, c_0) = \sum_{c \in C} |c - c_0| \cdot \frac{p(t_0, c)}{\sum_{t \in T} p(t, c)}, \quad (4)$$

where the probabilities are the predicted probabilities after the previous logistic regression step. Finally we predicted a label for each tweet according to the lowest loss, and calculated the mean sentiment for each entity. These were submitted as task E. We obtained a score of $EMD = 0.345$.

5 Review and Conclusions

In this paper we described our system of sentiment analysis adapted to participate in SemEval task 4. The highest ranking we reached was third place on the 5-label classification task. Compared with classification with 2 and 3 labels, in which we scored lower, and the fact we used similar workflow for tasks A, B, C, we speculate that the relative success is due to our sentiment treebank ranking on a 5-label basis. This can also explain the relatively superior results in quantification of 5 categories as opposed to quantification of 2 categories.

Overall, we have had some unique advantages and disadvantages in this competition. On the one hand, we enjoyed an additional twenty thousand tweets, where every node of the parse tree was labeled for its sentiment, and also had the manpower

to manually prune our dictionaries, as well as the opportunity to get feedback from our clients. On the other hand, we did not use any user information and/or metadata from Twitter, nor did we use the SemEval data for training the RNTN models. In addition, we did not ensemble our models with any commercially or freely available pre-trained sentiment analysis packages.

6 Future Work

We have several plans to improve our algorithm and to use new data. First, we plan to extract more semantic features such as verb and adverb classes and use them in neural network models as additional input. Verb classification was used to improve sentiment detection (Chesley et al., 2006); we plan to label verbs according to whether their sentiment changes as we change the tense, form and active/passive voice. Adverbs were also used to determine sentiment (Benamara et al., 2007); we plan to classify adverbs into sentiment families such as intensifiers ("very"), diminishers ("slightly"), positive ("delightfully") and negative ("shamefully").

Secondly, we can use additional data from Twitter regarding either the users or the entities-of-interest.

Finally, we plan to implement a long short-term memory (LSTM) network (Hochreiter and Schmidhuber, 1997) which trains on a sentence together with all the syntax and semantic features extracted from it. There is some work in the field of semantic modeling using LSTM, e.g. Palangi et al. (2014, 2016). Our plan is to use an LSTM module to extend the RNTN model of Socher et al. (2013) by adding the additional semantic data of each phrase and a reference to the entity-of-interest. An illustration of the computational graph for the proposed model is presented in figure 1. The inputs/outputs are: V is a word vector representation of dimension d, D encodes the parts-of-speech (POS) tagging, syntactic cate-
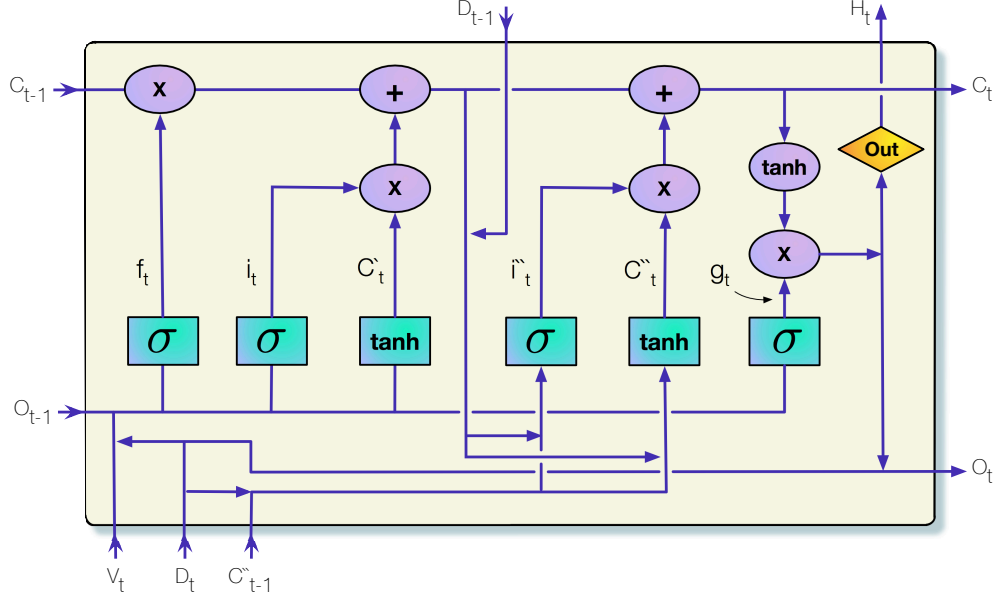
Figure 1: LSTM module; round purple nodes are element-wise operations, turquoise rectangles are neural network layers, orange rhombus is a dim-reducing matrix, splitting line is duplication, merging lines is concatenation.

gory and an additional bit indicating whether the entity-of-interest is present in the expression—all encoded in a 7 dimensional vector, C is a control channel of dimension d, O is an output layer of dimension $d + 7$ and H is a sentiment vector of dimension s.

The module functions are defined as following:

$$f_t = \sigma \left[L_f \left([V_t, D_t], O_{t-1} \right) \right]$$
$$i_t = \sigma \left[L_i \left([V_t, D_t], O_{t-1} \right) \right]$$
$$C'_t = \tanh \left[L_{C'} \left([V_t, D_t], O_{t-1} \right) \right]$$
$$i''_t = \sigma \left[L_{i''} \left([C''_{t-1}, D_t], [C_{t-1}, D_{t-1}] \right) \right]$$
$$C''_t = \tanh \left[L_{C''} \left([C''_{t-1}, D_t], [C_{t-1}, D_{t-1}] \right) \right]$$
$$g_t = \sigma \left[L_g \left([V_t, D_t], O_{t-1} \right) \right]$$
$$C_t = C_{t-1} \odot f_t + C'_t \odot i_t + i''_t \odot C''_t$$
$$H_t = W_{\text{out}} \cdot (g_t \odot \tanh(C_t))$$
$$O_t = [D_t, (g_t \odot \tanh(C_t))], \tag{5}$$

where $W_{\text{out}} \in \mathbb{R}^{s \times d}$ is a matrix to be learnt, $\odot$ denotes Hadamard (element-wise) product and $[.,.]$ denotes concatenation. The functions L_i are the six NN computations, given by:

$$L^k(S_{ij}) = S_{ij} T^{k,[1:d]} S_{ij}^\top + I_{0,0} W_{0,0}^k S_{ij}^\top$$
$$+ I_{0,1} W_{0,1}^k S_{ij}^\top + I_{1,0} W_{1,0}^k S_{ij}^\top$$
$$+ I_{1,1} W_{1,1}^k S_{ij}^\top$$
$$S_{ij} = ((v_i, s_i, e_i), (v_j, s_j, e_j)), \tag{6}$$

where (v_i, s_i, e_i) are the d dimensional word embedding, 6-bit encoding of the syntactic category and an indication bit of the entity-of-interest for the ith phrase, respectively, S_{ij} encodes the inputs of a left descendant i and a right descendant j in a parse tree and $k \in \{1, \ldots, 6\}$. Define $D = 2d + 14$, then $T^{[1:d]} \in \mathbb{R}^{D \times D \times d}$ is a tensor defining bilinear forms, $I_{I,J}$ with $I, J \in \{0, 1\}$ are indication functions for having the entity-of-interest on the left and/or right child and $W_{I,J} \in \mathbb{R}^{d \times D}$ are matrices to be learnt.

The algorithm processes each tweet according to its parse tree, starting at the leaves and going up combining words into expressions; this is different than other LSTM algorithms since the parsing data is used explicitly. As an example, figure 2 presents the simple sentence "Amobee is awesome" with its parsing tree. The leaves are given by d-dimensional word vectors together with their POS tagging, syntactic categories (if defined for the leaf) and an entity indicator bit. The computation takes place in the inner nodes; "is" and "awesome" are combined in a node marked by "VP" which is the phrase category. In terms of our terminology, "is" and "awesome" are the i, j nodes, respectively for "VP" node calculation. We define C''_{t-1} as the cell's state for the *left* child, in this case the "is" node. Left and right are concatenated as input V_t and the metadata D_t is from the *right* child while D_{t-1} is the metadata from the *left* child. The second calculation takes place at the
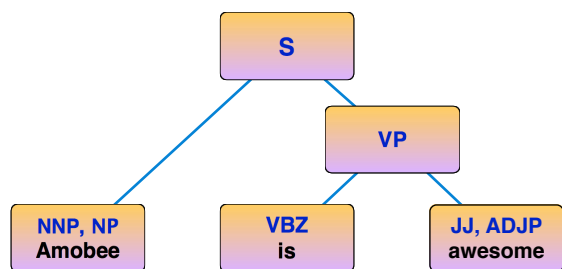
657

Figure 2: Constituency-based parse tree; the LSTM module runs on the internal nodes by concatenating the left and right nodes as its input.

root "S"; the input V_t is now a concatenation of "Amobee" word vector, the input O_{t-1} holds the O_t output of the previous step in node "VP"; the cell state C''_{t-1} comes from the "Amobee" node.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* .

Farah Benamara, Carmine Cesarano, Antonio Picariello, Diego Reforgiato Recupero, and Venkatramana S Subrahmanian. 2007. Sentiment analysis: Adjectives and adverbs are better than adjectives alone. In *ICWSM*. Citeseer.

Paula Chesley, Bruce Vincent, Li Xu, and Rohini K Srihari. 2006. Using verbs and adjectives to automatically classify blog sentiment. *Training* 580(263):233.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*. Association for Computational Linguistics, pages 28–34.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. pages 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.* 24(4):694–707. http://dl.acm.org/citation.cfm?id=2992449.2992457.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab K. Ward. 2014. Semantic modelling with long-short-term memory for information retrieval. *CoRR* abs/1412.6629. http://arxiv.org/abs/1412.6629.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Marta Recasens, Marie-Catherine de Marneffe, and Christopher Potts. 2013. The life and death of discourse entities: Identifying singleton mentions. In *North American Association for Computational Linguistics (NAACL)*.

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

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, et al. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*. Citeseer, volume 1631, page 1642.