A Comparative Study of Sentiment Analysis: State of the Art and New Proposals

Giulia Carocari, Jeff Giliberti, Filippo Graziano and Michele Marzollo
sentimental_sourdough
Department of Computer Science, ETH Zürich, Switzerland

***Abstract*—Sentimental analysis has attracted considerable attention in recent years for its wide range of applications and Twitter has emerged as a social media platform of major interest within this research area. The main contributions of this paper are: (1) investigating the effect of text-preprocessing on Twitter data (2) analysing the relationship between word embedding methods such as GloVe and Word2Vec and classification accuracy (3) comparing the performance of state-of-the-art models and present the application of techniques from other NLP and sequence modeling tasks to sentimental analysis.**

I. Introduction

Sentiment analysis is the task of identifying the sentiment or emotion associated with a certain text. It has many applications across multiple domains, such as consumer products or political elections, which lead it to become a particularly important branch of natural language processing [1].

Twitter is a microblogging platform that allows to collect a large amount of heterogeneous data on people's opinion [2]. The sentiment of a text can be extracted independently from the topic through the detection of emoticons representing positive and negative emotions [3]. The data used in this work is of this kind: we use 2.5M tweets that contained either the emoticon ":)" or ":("; the goal is to predict the emoticon originally present in the test data.

A large amount of studies on sentiment analysis is found in literature, such as [4], [5], [6]. This work contributes by comparing state of the art approaches and by proposing interesting combination of these models or suggesting new ideas to improve or extend them. More in detail, Section II first introduces the baselines against which we validate our experiments, then, it describes the algorithms we considered to perform preprocessing, word embeddings and classification. Section III presents the experimental results of the various approaches taken into account, which are analysed and compared to the baselines in Section IV. The paper focuses on three phases of our model's pipeline: the effect of different stages of text preprocessing, the comparison of a tailored implementation of word and tweet embeddings with pre-trained vectors and finally the evaluation of different choices or combinations of classifiers.

II. Models and Methods

A. Baselines

We compare our methods to two standard approaches of text classification.

The first one is to create sentence embeddings by performing the weighted sum of the word embeddings in the tweets [7] and then feed them to a standard classifier, such as Logistic Regression or Random Forests[1]. Considering the good results recently shown by neural networks, and to make the baselines comparable to our original models we use a feed-forward neural network for classification. The first layer performs a sum with learned weights of the word embeddings. Each of the two following layer has 256 hidden units and ReLU activations [8], and we finally add a layer with two units and softmax activation function, which gives as output the probability of belonging to each of the two classes. In the following, this model is referred to as MLP (Multi-Layer-Perceptron). We choose to use Stanford's pretrained embeddings of size 200 [9], trained on 2 billion tweets, which are state-of-the-art word representations[2].

The second baseline is a more advanced technique which is a standard approach of NLP [10]: we pass the sequence of embeddings to a Long-Short-Term-Memory cell [11] and pass the output to two fully-connected layers. More in detail, we use one LSTM layer with 256 cell units. The last output of the output sequence is fed to a feed-forward neural network equal to the one in the MLP. In both models we add dropout [12] with dropout rate 0.5 for regularization. Results are shown in the upper part of Table II.

B. Preprocessing

Several solutions have been proposed for text preprocessing in the context of sentiment analysis [13], [14], [15], some specifically on Twitter corpora [16]. The goal is to reduce noise in the dataset, thus reducing the size of the vocabulary, improving both training time and ideally the quality of the information provided to the model.

From the various approaches we extract four steps of text preprocessing, each applied on top of the previous one:

***Filtering** of duplicate tweets:* Approximately 10% of the tweets in the provided datset are duplicates. We remove them in order to obtain more truthful results on the validation accuracy score.

***Unfolding hashtags** and **Pattern-matching**:* for all tokens in the form "*#expression*" we infer the blanks in

[1]www.da.inf.ethz.ch/teaching/2020/CIL/files/exercises/exercise06.pdf

[2]Available for download at http://nlp.stanford.edu/data/glove.twitter.27B.zip

expression using a dynamic programming algorithm to determine the minimum cost split. Moreover, we perform pattern-matching (using GNU Sed regular expressions) to even out the writing style.

Spell-checking: Performed using aspell-python [17], a Python wrapper for GNU Aspell.

Lemmatization: Implemented by the WordNet [18] lemmatizer provided by the NLTK library [19]. Part of speech (POS) tags are retained in a separate file so that the information they provide can be reintroduced after the computation of word embeddings. This idea stems from [16], which outlines an imbalance in the frequency of certain types of verb tenses and adjective forms in positive and negative corpora.

C. Embeddings

To make it possible for the classification algorithms to "understand" natural language, we use two popular unsupervised word embedding algorithms: Word2vec [20] and GloVe [9].

For the former, we use the Gensim [21] implementation of the algorithm and set it to use the Skip-Gram model. Of the parameters it allows to set, we take into account: embedding size (d); min_count to set a threshold for the frequency of a word in the corpus so that it is inserted into the vocabulary; the size of the (symmetric) context window of each word ($context\_wdw$); the number of training epochs.

As regards GloVe, we employ our own implementation of the whole pipeline, keeping all the parameters listed above, with some further precautions.

When computing the cooccurrence matrix, we weight words in the context window according to the inverse of the relative distance between words.

For the actual training of the word embeddings, we follow the guidelines of the original authors, choosing $f(n) = \min(1, (n/100)^{3/4})$ as the weighting function of the model. Moreover, we use a Bold Driver [22] adaptive learning rate.

Finally, as anticipated in Section II-B, after the training step and if the lemmatized dataset is used, the possibility of appending POS information to the resulting word embedding is included. We encode each verb tense and each adjective form with a value, and concatenate these to the word embeddings.

D. Novel Solutions

We use several different neural networks to perform the classification task, some more classical and some more innovative. We tested different features and additions to improve performance. All models were implemented in TensorFlow 2.2.0 [23] with the Keras API. In particular, we either used custom Keras layers or built our own layers from scratch, so that our final models are sequential combinations of these layers. This makes the testing of different configurations very flexible.

All models end with a fully-connected network as in the baseline models, but where each of the first two layers has 512 hidden units.

The most commonly used architectures in NLP are RNNs and CNNs [4], so most of our networks are built on top of these two models. Moreover we also explored the effects of Attention [24], that automatically recognize which are the most relevant words in the sequence for our classification task.
The following are some of the models we evaluated. We give a name to each of the models for clarity. For implementation details, please refer to the code.

StackedLSTM: A common approach in NLP is to stack multiple recurrent layers to improve performance [5], [25]. In our case we stack two bidirectional LSTM layers [26] in order to capture dependencies from both previous and following words. Each bidirectional layer has 256 cell units and the two outputs of the forward and backward layers are concatenated.

CNN: We use a similar implementation to [6], which consists in applying 1-dimensional convolutions with different filter sizes on each sentence and then applying max-pooling over the feature maps. In addition to the paper, we tried different dilation factors on filters, obtaining slight improvements.

DoubleTCN: Temporal Convolutional Networks (TCNs) are convolutional neural networks that are used to model sequences. Our implementation is based on Bai et al. [27].

In the original paper the network is meant to produce an output sequence of the same length of the input sequence, where the output at each time step depends only on the past. In contrast, our network learns from both previous and following words, and if the output is followed by the final fully-connected layers we do not pad sequences in order to reduce the dimensionality. In this way, and by gradually reducing the number of filters, we obtain a much smaller output to be passed to the final layers.

Finally, we substitute ReLU activations suggested by [27] with Gated Linear Units [28] to boost performance. For the implementation we consulted both the original PyTorch implementation of the paper and [29].

Since tweets are short sequences, high dilation factors and large filters have a too large range, we stack two TCNs with filter size 3. The first TCN has dilation factor 1 and the second one has three layers with dilation factors 1,2,4.

TCN-RNN: Similarly to the StackedLSTM model we add a layer before the final LSTM. The role of the TCN layer is to extract higher level features from the sentence, to be passed to the LSTM layer. In this case we use TCN with a single layer equal to the first TCN in DoubleTCN. The LSTM is the same as in the StackedLSTM model. We also tested substituting the LSTM with the GRU [30].

TCN-CNN: As in TCN-LSTM, in order to increase the performance of the CNN, we add a TCN layer before it. The CNN equals the one in the CNN model.

Transformer: In Vaswani et al. [31], a new model (named Transformer) for sequence transduction based on attention mechanisms was introduced. Since our focus is only on a classification task, instead of the encoder-decoder architecture we limit our work to the encoder.

The model includes a sequence of hidden layers stacked on each other, each including two sub-layers: a multi-head self-attention layer followed by a feed-forward layer, both surrounded by residual connections. To allow for greater flexibility we replace the residual connection with highways [32].

Since the attention does not use any positional information, we also add positional encodings to the embeddings, as suggested in the original paper [31]; moreover, following [33], we do not only add them on the first layer, but we add separate trained positional encodings on each layer.

In order to flatten the sequence returned by the network to a single sentence embedding suitable for classification we reproduce [34] adding a trained [CLS] token at the beginning of the sequence.

Furthermore, stacking more than two layers is detrimental for performance, whereas, the use of many heads (e.g. 16) in the multi-head self-attention was beneficial. For the implementation we consulted [35].

RNN-Attention: In Yang et al. [36] attention was first used for document classification tasks; since we are interested in sentence level classification we simplify the model. The model stacks an attention layer on top of a bidirectional RNN to obtain a sentence token.

For the RNN we found that the LSTM performs slightly better than the GRU, which is the model suggested in the original paper [36]. For the attention layer, a simple attention layer gives better results than using more sophisticated multi-head attention layers or even a full Transformer.

E. Ensemble

Ensemble classification techniques can be adopted in order to achieve better classification performance as shown in [37] and [38]. [39] and [40] demonstrate that diversity and accuracy of individual classifiers are two crucial properties for ensemble learning. We combine diversity of text preprocessing and word embeddings as well as diversity of independent classifiers by collecting outputs of the softmax function of base classifiers and employ Support Vector Machine [41] to learn from them.

F. Experiments

Word embeddings were trained on the complete dataset (including the test set) for 50 epochs. We set $context\_wdw = 10$ and $min\_count = 10$, as they provide the best performance while allowing for reasonable training time. The quality of the embeddings is tested solely against the accuracy of the classification they provide. We do not rely on the classical evaluation metrics for word embeddings, such as linear substructures based on the semantics of words used by [9]. Nonetheless, the results obtained on the aforementioned metrics allow us to fix $d = 300$ for all experiments, as increasing the dimensionality of the embedding does not bring any significant advantage for the sentiment analysis task either.

Preprocessing type	Val. acc. (GloVe)		Val. acc. (Word2vec)	
	MLP	Stacked LSTM	MLP	Stacked LSTM
Filtering	**0.8307**	0.8531	0.8011	0.8285
Pattern Matching	0.8302	0.8541	0.8103	**0.8387**
Spell-checking	0.8297	**0.8545**	0.8125	0.8338
Lemmatization	0.8299	0.8503	0.8090	0.8333
Lemmatization + POS tags	0.8295	0.8512	**0.8138**	0.8364

Table I
CLASSIFICATION ACCURACY OF GLOVE AND WORD2VEC ($d = 300$) TRAINED ON VARIOUS PREPROCESSING LEVELS. CLASSIFIERS ARE TRAINED ON $\sim 180k$ TWEETS WITH 10000 VALIDATION TWEETS.

All neural networks were trained with the Adam optimizer [42] with learning rate $4 \cdot 10^{-4}$ and batch size 128. Data is splitted into training and validation, with 100 000 tweets in the validation set, and train the networks until early stopping. The validation predictions are also used to train the ensemble classifier. Models are trained for four epochs after reaching the lowest binary cross entropy loss on the validation set, and then restore the best weights.

For most models, training takes between 1.5 and 2.5 hours on a single Nvidia GeForce GTX 1080 Ti GPU and requires from 7 to 17 epochs; the Transformer instead trains in about 6 hours and for 29 epochs.

III. RESULTS

A. Preprocessing and Embeddings

We analyse the effects of preprocessing using a smaller dataset of approximately 190 thousand tweets (10000 of which are used for validation) on both embedding algorithms and two classifiers: the baseline MLP and the StackedLSTM. From Table I we evince that preprocessing mostly affects the classification performance of the MLP trained on Word2vec embeddings. The other models, on the other hand, benefit less from the refinement operated on the training data.

Table I also compares the quality of the produced word embeddings for the classification task. Word2vec performs worse than GloVe on both classifiers, although, it is also the algorithm that most benefits from the preprocessing steps.

B. Classifiers

The results of baselines and novel models are reported in Table II. There is some small variability in results between runs due to the stochasticity of Keras training on GPUs.

Overall, we found that for this task deeper networks are not necessary; increasing the depth typically results in a degradation of performance.

From our experiments, we observe that RNNs behave generally better than CNNs. Among convolutional neural networks, Temporal Convolutions perform significantly better than classical ones, obtaining accuracies near to the ones of LSTM. Adding a TCN layer before both LSTM and CNN layers shows an increase in performance: TCN+LSTM obtains slightly better results than StackedLSTM. Encouraging results were also achieved by substituting the LSTM after the TCN with a GRU. In constrast, increasing network sizes of CNNs and LSTMs is not as beneficial as adding a TCN layer. Ultimately, Transformer and the other attention-related mechanisms are less advantageous than stacking two LSTMs.

IV. Discussion

The inclusion of the StackedLSTM in the analysis of preprocessing techniques allows us to give a more comprehensive picture than the one provided by the Naïve Bayes and SVM classifiers presented in Haddi et al. [13] and Pak et al. [16]. Indeed, from Table I we deduce that text preprocessing is mostly useful when the model is less powerful and starts from a lower baseline accuracy on the original dataset, i.e. the MLP trained using Word2vec. With more refined embeddings and classification algorithms, on the other hand, aggressive preprocessing techniques tend to be detrimental.

Regarding the difference in performance caused by word embeddings, we trace one of the reasons for this back to the rather small corpus size. In fact, Word2vec's strength relies on its efficiency in handling large corpus sizes [20]. However, the reduction of the vocabulary size through preprocessing is likely to increase the amount of information the algorithm extracts from the available tweets, thus increasing classification accuracy.

NN type	Num. Parameters	Val. Accuracy	Test Accuracy
MLP	282k	0.8362	0.8426
BaselineLSTM	600k	0.8661	0.8700
MLP	195k	0.8372	0.8550
StackedLSTM	3.2M	**0.8811**	0.8856
CNN	1.9M	0.8678	0.8780
DoubleTCN	3.8M	0.8765	0.8846
TCN-LSTM	3.7M	**0.8811**	0.8862
TCN-GRU	3.3M	0.8791	**0.8888**
TCN-CNN	4.1M	0.8702	0.8794
Transformer	6.6M	0.8725	0.8798
LSTM-Attention	5.2M	0.8782	0.8886
Ensemble	-	0.8907[3]	0.8978

Table II
Classification results of baseline and novel classifiers.

	Stack.LSTM	CNN	DoubleTCN	TCN-LSTM	TCN-CNN	Transformer	LSTM-Att.
Stack.LSTM	-	2.9	3.0	2.8	3.1	2.6	2.5
CNN	4.2	-	3.9	4.2	3.9	3.7	4.1
DoubleTCN	3.3	3.0	-	2.9	2.8	3.3	3.4
TCN-LSTM	2.8	2.9	2.6	-	2.8	2.9	2.9
TCN-CNN	3.6	3.1	3.0	3.3	-	3.4	3.4
Transformer	3.5	3.3	3.8	3.7	3.8	-	3.4
LSTM-Att.	2.8	3.1	3.3	3.1	3.2	2.9	-

Table III
Percentage of validation tweets correctly labelled by the column classifier but wrongly predicted by the row classifier.

The comparison between the baseline-MLP and the MLP trained on the preprocessed dataset outlines how moderate preprocessing and a tailored implementation of GloVe embeddings yield an increase in accuracy while using fewer parameters (due to the smaller vocabulary size), thus reducing training time for the networks.

As for the second baseline, the advanced models present an increase of the classification accuracy of approximately 1.5% over the state-of-the-art baseline. As mentioned in Section III, an increase in model complexity does not always lead to a comparable improvement in performance. It is more beneficial to use networks tailored to the task (e.g. TCN-GRU) rather than increasing the number of parameters of less suited classifiers.

The training of a larger number of different classifiers is nonetheless quite effective; in Table III, we compare the percentage of validation tweets that are incorrectly labelled by one classifier but classified correctly by the other. Intuitively, it is clear that models with higher accuracy will also perform better in this comparison; however, the data also shows how a weaker classifier can learn patterns missed by more complex models which is effective in the presence of an ensemble, which increases by 1% the stand-alone classifier accuracy[3].

V. Summary

We have shown that text-preprocessing is beneficial only for weak networks, whereas, the choice of expressive word embedding is critical for classifier learning. Finally, most of the state-of-the-art solutions provide comparable results to which ensemble methods are applied in order to improve the overall accuracy.

Acknowledgements

We thank ETH Zürich for giving us the availability of the computational resources of the Leonhard cluster.

[3]Obtained through 5-fold cross-validation on the original validation set.

References

[1] B. Liu, "Sentiment analysis and opinion mining," *Synthesis lectures on human language technologies*, vol. 5, no. 1, pp. 1–167, 2012.

[2] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. J. Passonneau, "Sentiment analysis of twitter data," in *Proceedings of the workshop on language in social media (LSM 2011)*, 2011, pp. 30–38.

[3] J. Read, "Using emoticons to reduce dependency in machine learning techniques for sentiment classification," in *Proceedings of the ACL student research workshop*, 2005, pp. 43–48.

[4] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative study of cnn and rnn for natural language processing," 2017.

[5] K. Cho, B. van Merriënboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014.

[6] Y. Kim, "Convolutional neural networks for sentence classification," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014.

[7] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," 2016.

[8] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010.

[9] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[10] J. Chung, Çaglar Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *ArXiv*, vol. abs/1412.3555, 2014.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 1997.

[12] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[13] E. Haddi, X. Liu, and Y. Shi, "The role of text pre-processing in sentiment analysis," *Procedia Computer Science*, vol. 17, p. 26–32, 2013.

[14] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques," *CoRR*, vol. cs.CL/0205070, 2002.

[15] K. Dave, S. Lawrence, and D. M. Pennock, "Mining the peanut gallery: Opinion extraction and semantic classification of product reviews," in *Proceedings of the 12th International Conference on World Wide Web*, ser. WWW '03. New York, NY, USA: Association for Computing Machinery, 2003, p. 519–528.

[16] A. Pak and P. Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining," vol. 10, 2010.

[17] W. Muła, "aspell-python," https://github.com/WojciechMula/aspell-python, last access July 24, 2020.

[18] P. Oram, "Wordnet: An electronic lexical database. christiane fellbaum (ed.). cambridge, ma: Mit press, 1998. pp. 423." *Applied Psycholinguistics*, vol. 22, no. 1, p. 131–134, 2001.

[19] "Natural language toolkit," http://www.nltk.org/, last access July 24, 2020.

[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[21] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, 2010, pp. 45–50.

[22] R. Battiti, "Accelerated backpropagation learning: Two optimization methods," *Complex systems*, vol. 3, no. 4, pp. 331–342, 1989.

[23] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[24] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2015.

[25] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," 2013.

[26] M. Schuster, K. K. Paliwal, and A. General, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, 1997.

[27] S. Bai, J. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv:1803.01271*, 2018.

[28] Y. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," 2016.

[29] "Tf tcn," https://github.com/YuanTingHsieh/TF_TCN, last access September 11, 2020.

[30] K. Cho, B. van Merriënboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014.

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[32] R. K. Srivastava, K. Greff, and J. Schmidhuber, “Highway networks,” 2015.

[33] R. Al-Rfou, D. Choe, N. Constant, M. Guo, and L. Jones, “Character-level language modeling with deeper self-attention,” in *Thirty-Third AAAI Conference on Artificial Intelligence*, 2019. [Online]. Available: https://arxiv.org/abs/1808.04444

[34] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” 2018.

[35] “The transformer model in attention is all you need: a keras implementation.” https://github.com/Lsdefine/attention-is-all-you-need-keras, last access September 11, 2020.

[36] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, “Hierarchical attention networks for document classification,” 2016, pp. 1480–1489.

[37] K. Tumer and J. Ghosh, “Error correlation and error reduction in ensemble classifiers,” *Connection Science*, vol. 8, 08 1996.

[38] G. Brown, J. Wyatt, R. Harris, and X. Yao, “Diversity creation methods: a survey and categorisation,” *Information Fusion*, vol. 6, no. 1, pp. 5 – 20, 2005, diversity in Multiple Classifier Systems. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1566253504000375

[39] Z.-H. Zhou, J. Wu, and W. Tang, “Ensembling neural networks: Many could be better than all,” *Artificial Intelligence*, vol. 137, pp. 239–263, 04 2002.

[40] L. Yang, “Classifiers selection for ensemble learning based on accuracy and diversity,” *Procedia Engineering*, vol. 15, pp. 4266–4270, 12 2011.

[41] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[42] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *International Conference on Learning Representations*, 2014.

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

A COMPARATIVE STUDY OF SENTIMENT ANALYSIS: STATE OF THE ART AND NEW PROPOSALS

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
CAROCARI	GIULIA
GILIBERTI	JEFF
GRAZIANO	FILIPPO
MARZOLLO	MICHELE

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zurich, July 29, 2020	[illegible]
	[illegible]
	[illegible]
	[illegible]

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.