

Applying Ensembling Methods for Sentiment Classification of Tweets

Dimitrios Lekkas, Foteini Strati, Andreas Triantafyllos, Emmanouil Vardas

Group: **Freddo Espresso Sketo**, Department of Computer Science, ETH Zurich, Switzerland

Abstract—Sentiment analysis techniques have been widely used in Twitter to automatically extract emotions and opinions within tweets. In this work we consider several models for text classification, including classical techniques based on word embeddings and classifiers, as well as complex recurrent neural network architectures, and state-of-the-art pretrained transformer models. Our final proposal combines the most accurate of these models in an optimal way and manages to achieve higher accuracy than each individual model.

I. INTRODUCTION

Sentiment analysis is a field of Natural Language Processing (NLP) used to detect polarity, feelings, opinions and intentions within text. It is widely applied in areas such as social media monitoring [1], customer experience surveys [2], healthcare [3] and recommender systems ([4], [5]). Since microblogging and text messaging are the main media of communication nowadays, there has been an enormous evolution in the field of sentiment analysis attempting to extract information in an automatic and highly accurate way.

Twitter is one of the microblogging platforms that gained a lot of attention over the past few years. With a huge amount of users expressing their thoughts and preferences through Twitter, many companies and media organizations use sentiment analysis tools to extract people’s opinions and reviews about their products. The variety of topics for which users tweet, the informal language and the characteristics of speech used in tweets such as emoticons, colloquial expressions, abbreviations, are some of the challenges that need to be faced when applying sentiment analysis techniques in tweets.

The approaches used for sentiment analysis break down into three primary categories [6], namely lexicon-based (or knowledge-based) techniques ([7], [8]), machine learning methods ([9], [10], [11], [12]), and hybrid approaches [13]. Lexicon-based approaches relate words with emotions and classify text based on the presence of these words. Instead, machine learning approaches utilize ML models and techniques to extract features from text, while hybrid approaches leverage both worlds.

In this project, we experiment with various machine learning models for text classification. We implement as baselines a Support Vector Machine as well as an XG-Boost classifier and evaluate them using both our own and pretrained embeddings. Thereafter, we experiment with a Bidirectional LSTM-based model and observe a significant increase at the accuracy of our predictions. Afterwards, we utilize pretrained state-of-the-art transformers such as

BERT [9] and RoBERTa [10] and as a final step we perform ensembling of our best models with a variety of blending techniques. Doing so, we achieve an absolute improvement in accuracy compared to each individual model.

II. MODELS AND METHODS

A. Preliminaries

We are given a set of 2.5 million tweets, from which the one half is labeled as positive (1) and the rest as negative (−1). The train and validation sets arise from the split of this dataset. Our final model is evaluated, in terms of classification accuracy, on a set that consists of 10k tweets.

B. Preprocessing of Tweets

We experiment with several variants of preprocessing the given data, in order to observe the impact of different ways of data preparation to the final result. Since the tweets are already in lowercase, we decide that the first dataset should consist of the raw data (*‘raw’*). For the second version, we remove the punctuation in each tweet (e.g. !?-_&”) and replace two or more subsequent spaces with a single one (*‘punct’*). The third stage of preprocessing additionally replace emoticons (e.g. :-), <3) with the general sentiment that they represent (<happyface>, <heart> respectively) (*‘punct-emojis’*). Afterwards, we replace the slang expressions that exist in the tweets with the respective meaning (e.g. bday-birthday, dwi-deal with it) and we constrain the letter repetitions in words to two letters (e.g. sooo glaaaaaad-soo glad). By doing so, we preserve the emphasis that was given to words and expressions written in this way. In this version, we also correct the contractions (e.g. haven’t-have not) and remove the numbers and any word that would not begin with a letter from the alphabet (*‘punct-emojis-slang’*). The final version of preprocessing combines the previous techniques and further removes the most common stopwords of the English language (e.g. I, myself, where) (*‘punct-emojis-slang-stopwords’*). For our baselines, we make use only of *‘punct-emojis’*, added with removing duplicate tweets.

C. Word Embeddings

A word embedding is a latent mapping of a textual word in a real-value vector space. The goal is to capture semantic relations between words through distances or angles of the respective vectors. Here, we make extensive use of the GloVe model [14]. Learning GloVe word embeddings is an unsupervised learning task that takes into account the co-occurrence counts of words in a given corpus, captured

by the co-occurrence matrix $N = (n_{ij}) \in \mathbb{N}^{|V| \times |C|}$, where n_{ij} is the number of occurrences of word $w_i \in V$ in the context of word $w_j \in C$. The objective is to minimize the following weighted squared loss:

$$H(\theta; N) = \sum_{i,j} f(n_{ij})(x_i^\top y_j + b_i + c_j - \log n_{ij})^2$$

, where a popular used function is $f(n) = \min(1, (\frac{n}{n_{max}})^a)$, with $a \in (0, 1]$ and n_{max} is used as threshold for the influence of large counts. The parameters of our problem consist of $\theta = (X, Y)$ where $X = [x_{w_1}, \dots, x_{w_{|V|}}] \in \mathbb{R}^{|d| \times |V|}$ and $Y = [y_{w_1}, \dots, y_{w_{|C|}}] \in \mathbb{R}^{|d| \times |C|}$, assuming d -dimension embeddings with b_{w_i} and c_{w_j} being integrated in x_{w_i} and y_{w_j} accordingly [14].

The loss is a non-convex function, hence gradient-based methods can only converge to a local minimum. Methods like Gradient Descent cannot be applied here, because computation of gradient involves a large number of parameters, hence it is computationally expensive. A very common technique is to apply cheap updates by using Stochastic Gradient Descent (SGD) for all non-zero entries of co-occurrence matrix, for a number of epochs.

1. $x_i^{new} = x_i + 2\eta f(n_{ij})(\log n_{ij} - x_i^\top y_j)y_j$
2. $y_j^{new} = y_j + 2\eta f(n_{ij})(\log n_{ij} - x_i^\top y_j)x_i$

D. Pretrained Embeddings

Pretrained word embeddings is a commonly used technique in modern NLP systems and when used they usually perform better than embeddings learned from scratch. We make use of the Twitter embeddings dataset provided in [15], which includes embeddings of 50, 100 and 200 dimensions for a vocabulary of 1.2 million words extracted from 2 billion tweets and test many different models with or without these pretrained embeddings.

E. Baselines

We provide baseline implementations that differ in terms of the classifier that we use, i.e. Support Vector Machine and XGBoost and in terms of manually constructed or pretrained word embeddings. The input of the classifiers consists of the embedding representation of tweets provided in our dataset. Especially, with $\eta = 0.001$ and $a = \frac{3}{4}$ (drawn empirically) we train our own GloVe embeddings with 50, 100 and 200 dimensions by running the SGD algorithm for the co-occurrence matrix produced from the vocabulary of our dataset for 100 epochs, while also producing a fourth version of embeddings with 200 dimensions and 200 epochs. We also get results for using pretrained word embeddings for the words that belong in the intersection of vocabularies of our dataset and the pretrained embeddings Twitter dataset.

Each word w_i is finally represented by one embedding vector $z_i = x_i + y_i$. It is important to note that in our case, the co-occurrence matrix is symmetric and that $V = C$, hence x_i and y_i are different only because they are

randomly initialized with values coming from the normal distribution. Finally, the tweet embedding representation is drawn as the average of the word embeddings that belong to each tweet.

1) **Support Vector Machine:** We make use of a Support Vector Machine with linear kernel and default parameters, whose regularization parameter C has been fine tuned through a grid search using 5-fold cross validation. Our results can be seen on the first four lines of Table I.

2) **XGBoost:** We receive results for a gradient boosting algorithm like a tree-based XGBoost classifier with fine tuned parameters of max depth, learning rate and number of estimators through a 5-fold cross validation. Our results can be found on the last two lines of Table I. Because the training time of such models grows with the change of depth and number of estimators, we deliberately draw results only for the pretrained versions of our data, that achieve better accuracy.

One major problem of the baseline methods is that the computation of embeddings is decoupled from the classification objective: a classifier is fed with finalised embeddings that have been trained under an unsupervised learning task, and produces an output without any interference in the embedding computation process. Moreover, averaging word embeddings to produce the final vector representation of a tweet is a simplistic approach that does not consider significant aspects like word order in tweets, double negations, resulting in positive outcome etc. Hence, there is a need for more complicated models that tackle the above drawbacks.

F. LSTM-based Model

Recurrent Neural Networks (RNNs) have been studied extensively and have been a popular choice for NLP problems. Their sequential processing nature closely resembles the way humans process text information and they weigh on the ordering of the words which is a crucial feature for classifying sentiment. Though, training standard RNNs to solve problems that require learning long-term temporal dependencies is challenging since the gradient of the loss function decays exponentially with time [16]. On the other hand, LSTMs have a ‘memory cell’ that can maintain information in memory for long periods of time which renders them a better choice for capturing longer-sentence dependencies.

We start by applying preprocessing on the sentences and we pass them to an embedding layer which uses the pretrained GloVe word vectors. Afterwards, we feed those vectors to a bidirectional LSTM, which summarizes information from both directions. Specifically, the bidirectional LSTM first processes the sentence as usual and then also processes it backwards and produces the final annotations which are represented by the concatenation of both vectors. We now have representations of the sentence that focus on each word separately and we insert those representations to a dense network with a softmax activation function. This dense layer plays the role of creating a weighted

average that is supposed to learn the proper weights for each word to simulate an attention mechanism which finds the relative contribution of each word. Finally, we apply dropout regularization to avoid over-fitting and we have our final layer which connects us to the output (i.e binary prediction).

G. Pretrained Models

The pretrained models are networks that have already undergone training on large datasets on objectives that may be different from the final purpose that they will be used for. Due to time restraints and/or computational costs, pretrained models is a commonly used technique in modern NLP systems. Pretrained language representations can be applied in two ways: **feature-based** and **fine-tuning**. In feature-based the model augments the features used by exploiting the pretrained representations. In fine-tuning one or more layers are inserted and the additional parameters along with the pretrained ones are fine-tuned for a specific task. In an attempt to alleviate the limitations of LSTM, we use the latter strategy and experiment with BERT model and a variation of it, named RoBERTa.

BERT [9] is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. Devlin et al. in [9] argue that the major limitation of techniques used before BERT is that the standard language models are unidirectional and this negatively affects performance on tasks where it is crucial to incorporate context from both directions. BERT manages to overcome this unidirectionality constraint by using a “masked language model” (MLM) pretraining objective. The masked language model randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked token. In addition to the masked language model, the model is also pretrained on a “next sentence prediction” (NSP) task.

BERT is a multi-layer bidirectional transformer encoder (as introduced by Vaswani et al. in [17]) with L layers. Each block uses A self-attention heads with hidden dimension H . For this project we use two models with different sizes: the BERT_{BASE} where $L=12$, $H=768$, $A=12$ and BERT_{LARGE} where $L=24$, $H=1024$, $A=16$ which correspond to 110M and 340M parameters respectively. BERT accepts a constrained input of 512 tokens (words, numbers or punctuation marks). However this is not a problem for us since the maximum number of tokens in our dataset does not exceed 100 tokens. We fine-tune both of the aforementioned models with batch size 64, learning rate $2 \cdot e^{-5}$, warmup training proportion 10% and maximum of 3 epochs.

Afterwards, inspired by the work of Sun et al., [18] we further pretrain both BERT models on the given data set and observe the impact of the additional pretraining on task specific data. We retrain BERT model with batch size 64, learning rate $5 \cdot e^{-5}$, total training steps 10000 and warmup proportion 10%. We fine-tune the generated model keeping the same batch size, learning rate $2 \cdot e^{-5}$, 4 epochs and using

the 4 last layers of the model as feature for classification according to [18].

Liu et al. [10] found out that BERT was significantly undertrained, and they managed to design and train a more robust version of it (RoBERTa) which could match or exceed the results of its competitors. In order to achieve this, RoBERTa was trained on a new larger dataset with batches of bigger size and for more epochs. Moreover, the next sentence prediction objective was removed and longer sequences were used, enabling the model to learn long-range dependencies.

Therefore, after training the BERT models, we fine-tune RoBERTa models which are based on BERT_{BASE} and BERT_{LARGE}. We also experiment with fusing RoBERTa and LSTM networks. Inspired by [19] we add a BiLSTM layer on top of RoBERTa and measure its impact in terms of classification accuracy. All RoBERTa models are best fine-tuned with batch size 64 and learning rate $1 \cdot e^{-5}$.

H. Ensembling

Ensembling techniques have been widely used in sentiment analysis to exploit the strengths and adjust the weaknesses of individual models [9], [20], [21], [22]. In an attempt to observe the impact of combining different approaches in our final predictions, our last step is to ensemble some of the most accurate models we developed. These are namely the BiLSTM-base model, the fine-tuned BERT_{BASE}, BERT_{LARGE}, RoBERTa_{BASE} and RoBERTa_{LARGE} models, and the RoBERTa_{LARGE} with a BiLSTM layer on top of it.

We split the provided dataset of 2.5M tweets into a training set S_{train} , which consists of 90% of the whole dataset and validation set S_{val} consisting of the remaining 10% of the data. The aforementioned models are trained using the S_{train} dataset and validated in S_{val} . The S_{val} dataset is utilized in the ensembling procedure to learn the optimal combination of the individual models.

For combining our models, we use both a Linear Regressor and an XGBoost classifier:

1) **Linear Regressor:** Let p_{ij} denote the prediction probability of the individual model i that the j sample of S_{val} belongs to class -1 (negative sample). Then, we need to find the optimal weights w_i , such that the final prediction of sample j belonging in class -1 is given by the weighted sum:

$$p_j = w_0 + \sum_i w_i \cdot p_{ij}$$

In order to find the optimal weights w_i we use a linear regression model, having as input a matrix $\mathbf{X} \in \mathbb{R}^{|S_{val}| \times N}$, where N is the number of individual models, and producing a matrix $\mathbf{y} \in \mathbb{R}^{|S_{val}|}$ containing the predictions for each sample. We fit the regressor to the validation set S_{val} and obtain the best threshold value p_{th} , such that if $p_j \leq p_{th}$, the tweet j belongs in class -1, else it belongs in class 1, by maximizing the validation accuracy of the regressor.

2) **XGBoost**: The second ensembling procedure is to apply the XGBoost classifier over the matrix of the prediction probabilities $\mathbf{X} \in \mathbb{R}^{|S_{val}| \cdot 2 \cdot N}$ generated by the N individual models for the set S_{val} . We apply 10-fold cross validation to the XGBoost classifier in order to find its optimal parameters and select the best model according to the mean attained validation accuracy over the 10 folds.

III. RESULTS

At this and the following section we present our most meaningful experiments and conclusions.

Model _{dataset-embeddings}	50d	100d	200d	200d-200epochs
SVM _{raw tweets}	0.6293	0.6264	0.6275	0.6394
SVM _{raw-pretrained}	0.7245	0.7569	0.7725	0.7743
SVM _{preprocessed tweets}	0.6007	0.6005	0.5993	0.6127
SVM _{preprocessed-pretrained}	0.7178	0.7462	0.7626	0.7641
XGBoost _{raw-pretrained}	0.7820	0.7991	0.8037	0.8045
XGBoost _{preprocessed-pretrained}	0.7598	0.7760	0.7801	0.7824

Table I

BASELINES CROSS-VALIDATION ACCURACY FOR DIFFERENT INPUTS

1) **Baselines**: For our baselines, we conclude that pre-trained embeddings provide better accuracy results in comparison to the manually crafted ones. Intuitively, such embeddings have been trained on a very large corpus of tweets that has the potential to provide a more accurate semantic representation of each word. At this setting, the growing number of dimensions improves the accuracy, as the relations between words are better captured in a vector space of larger dimensions. In contrast, the impact of dimensions is not apparent in our own embeddings. This can be explained by the fact that for the given size of our dataset, SGD needs more than 100 epochs for embeddings to converge to their final values. For our own embeddings, 200 epochs lead to better results, whereas accuracy from pretrained ones appears to be more robust in terms of epochs. Overall, the use of pretrained embeddings outperforms embeddings trained from scratch both in terms of computation time and quality of representation.

2) **Impact of data preprocessing**: In order to visualize the impact of different preprocessing techniques we used, we present at Table II the attained validation accuracy of our BiLSTM-based and RoBERTa model. We split the large dataset of 2.5M tweets into training and validation set, with the validation set consisting 5% of the whole dataset.

Preprocessing	Validation Accuracy	
	BiLSTM-based	RoBERTa
raw	0.8815	0.9163
punct	0.877	0.9039
punct-emojis	0.8715	0.9037
punct-emojis-slang	0.8717	0.8996
punct-emojis-slang-stopwords	0.8592	0.882

Table II

IMPACT OF DIFFERENT PREPROCESSING ON BiLSTM-BASED AND PRETRAINED ROBERTA MODEL

3) **Best individual models and ensembling**: As a final part, we present the accuracy achieved by our best individual models as well as by the two ensembling techniques that we utilized. The standalone models are trained at the 90% of the dataset, named S_{train} and evaluated at the rest 10%, named S_{val} . S_{val} is used for the training of the ensemble models. For completeness purposes, we also present the accuracy obtained at the public test set on Kaggle.

Model	Validation Accuracy	Public Test Accuracy
Individual Models		
BiLSTM-based model	0.8813	0.8766
BERT _{BASE}	0.9013	0.8988
BERT _{LARGE}	0.9046	0.90040
BERT _{LARGE-retrained}	0.9048	0.90640
RoBERTa _{BASE}	0.9085	0.9078
RoBERTa _{LARGE}	0.91546	0.913
RoBERTa _{LARGE} + BiLSTM	0.91582	0.9104
Ensembled Models		
Linear Regression Model		0.915
XGBoost		0.916

Table III

ACCURACY ACHIEVED BY THE BEST INDIVIDUAL MODELS AND ENSEMBLING

IV. DISCUSSION

Comparing Table I and Table II, it is obvious that all the models which are presented significantly outperform the two baseline models. Furthermore, we observe that pretrained models (BERT and RoBERTa) achieve higher accuracy over LSTM based models. Adding a BiLSTM on top of RoBERTa does not seem to offer improvement compared to the simple RoBERTa. Preprocessing the tweets that were given as data and test sets is not beneficial and we believe that this happens due to information (exclamation and question marks, emoticons) that is deprived by this process. Even for simple baselines, we attribute this behavior to the removal of duplicates that comprises $\sim 10\%$ in both positive and negative tweets, affecting co-occurrence counts.

Moreover, it is evident that using ensembling techniques increases the predictive ability of the final model since they combine effectively the merits of the different classifiers. More specifically, XGBoost manages to achieve better results in comparison with Linear Regressor and hence we select this as our final model.

V. SUMMARY

For the purposes of this project, we approach the Text Sentiment Classification problem in microblogging using a variety of deep learning models as well as two baseline models. We consider a LSTM which is a recurrent model and transformer models (BERT, RoBERTa) outperforming the two baseline models (SVM, XGBoost). Finally, we exploit two ensemble methods to combine individual models and further increase the accuracy of our final model.

REFERENCES

- [1] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau, "Sentiment analysis of twitter data," in *Proceedings of the Workshop on Languages in Social Media*, ser. LSM '11. USA: Association for Computational Linguistics, 2011, p. 30–38.
- [2] D. Gräbner, M. Zanker, G. Fliedl, and M. Fuchs, "Classification of customer reviews based on sentiment analysis," in *Information and Communication Technologies in Tourism 2012*, M. Fuchs, F. Ricci, and L. Cantoni, Eds. Vienna: Springer Vienna, 2012, pp. 460–470.
- [3] L. Abualigah, H. E. Alfar, M. Shehab, and A. M. A. Hussein, *Sentiment Analysis in Healthcare: A Brief Review*. Cham: Springer International Publishing, 2020, pp. 129–141. [Online]. Available: https://doi.org/10.1007/978-3-030-34614-0_7
- [4] S. Kumar, S. S. Halder, K. De, and P. P. Roy, "Movie recommendation system using sentiment analysis from microblogging data," 2018.
- [5] R. Jayashree and D. Kulkarni, "Recommendation system with sentiment analysis as feedback component," in *Proceedings of Sixth International Conference on Soft Computing for Problem Solving*, K. Deep, J. C. Bansal, K. N. Das, A. K. Lal, H. Garg, A. K. Nagar, and M. Pant, Eds. Singapore: Springer Singapore, 2017, pp. 359–367.
- [6] A. D'Andrea, F. Ferri, P. Grifoni, and T. Guzzo, "Approaches, tools and applications for sentiment analysis implementation," *International Journal of Computer Applications*, vol. 125, pp. 26–33, 09 2015.
- [7] A. Jurek, M. D. Mulvenna, and Y. Bi, "Improved lexicon-based sentiment analysis for social media analytics," *Security Informatics*, vol. 4, no. 1, p. 9, Dec 2015. [Online]. Available: <https://doi.org/10.1186/s13388-015-0024-x>
- [8] C. Musto, G. Semeraro, and M. Polignano, "A comparison of lexicon-based approaches for sentiment analysis of microblog," *CEUR Workshop Proceedings*, vol. 1314, pp. 59–68, 01 2014.
- [9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018.
- [10] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019.
- [11] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and B. Xu, "Text classification improved by integrating bidirectional lstm with two-dimensional max pooling," 2016.
- [12] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques," *EMNLP*, vol. 10, 06 2002.
- [13] I. Gupta and N. Joshi, "Enhanced twitter sentiment analysis using hybrid approach and by accounting local contextual semantic," *Journal of Intelligent Systems*, vol. 29, 09 2019.
- [14] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *EMNLP*, vol. 14, 2014, pp. 1532–1543.
- [15] <https://nlp.stanford.edu/projects/glove/>, "Glove: Global vectors for word representation," accessed: 31-07-2020.
- [16] S. Hochreiter, A. S. Younger, and P. R. Conwell, "Learning to learn using gradient descent," in *Proceedings of the International Conference on Artificial Neural Networks*, ser. ICANN '01. Berlin, Heidelberg: Springer-Verlag, 2001, p. 87–94.
- [17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.
- [18] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to fine-tune bert for text classification?" 2019.
- [19] A. Thorne, Z. Farnsworth, and O. Matus, "Research of lstm additions on top of squad bert hidden transform layers."
- [20] T. Dadu, K. Pant, and R. Mamidi, "Bert-based ensembles for modeling disclosure and support in conversational social media text," 2020.
- [21] Y. Xu, X. Qiu, L. Zhou, and X. Huang, "Improving bert fine-tuning via self-ensemble and self-distillation," 2020.
- [22] C. Xu, S. Barth, and Z. Solis, "Applying ensembling methods to bert to boost model performance."

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

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):

Applying Ensembling Methods for Sentiment Classification of Tweets

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):

Lekkas

Strati

Triantafyllos

Vardas

First name(s):

Dimitrios

Foteini

Andreas

Emmanouil

With my signature I confirm that

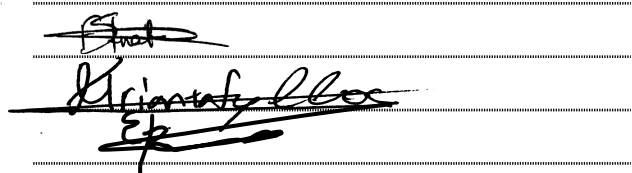
- I have committed none of the forms of plagiarism described in the '[Citation etiquette](#)' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

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

Place, date

Zurich, 31.07.2020

Signature(s)


Dimitrios Lekkas

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