

Profiling Irony and Stereotype Spreaders with Language Models and Bayes' Theorem

Xinting Huang

University of Copenhagen, Denmark

Abstract

The goal of profiling irony and stereotype spreaders is to classify authors as irony spreaders or not, based on a certain number of their tweets. In this paper, we present our novel system, which is different from typical approaches to classification tasks. Instead of extracting features and training machine learning classifiers, we exploit the Bayes' theorem and the property and function of language models to derive a classification system. We explain in detail why our system is able to make the right predictions, and also explore the characteristics of our new system by conducting further experiments. Finally, experimental results show that our approach can effectively classify ironic and non-ironic users and achieve a good performance on the shared task IROSTEREO.

Keywords

author profiling, irony detection, language models, Bayes' theorem

1. Introduction

Social media plays a more and more important role in our daily life. It has a lot of advantages for us, but it is also a hotbed of offensive and aggressive speech. It is therefore of great significance to profile users on social media. In the task Profiling Irony and Stereotype Spreaders on Twitter (IROSTEREO) 2022 [1, 2], we focus on profiling ironic authors on Twitter, especially those who use irony to spread stereotypes. The goal is to classify users on Twitter as ironic or not, given a certain number of their tweets. This task is to deal with the subtlety and complexity of human language. When using irony, language can be used to mean something opposite to its literal meaning. Moreover, the stereotype can be well hidden, which requires sufficient knowledge of human society and a good understanding of the context to be detected. Thus this task is somewhat challenging.

In this paper, we present our solution for the IROSTEREO task. We propose a novel approach to classify the users with their tweets. Instead of extracting features and training a classifier to explicitly do classification on the data, we make use of the function of language models and apply Bayes' Theorem to make predictions. In this way, we avoid selecting and extracting features manually, which is quite necessary when doing classification with very long text inputs. Experimental results have shown that our approach is effective on the IROSTEREO task. It can classify users as ironic or not with high probability. Our approach achieves 92.78% accuracy on


Code is available at <https://github.com/huangxt39/BayLMs>

CLEF 2022: Conference and Labs of the Evaluation Forum, September 5–8, 2022, Bologna, Italy

✉ wbn969@alumni.ku.dk (X. Huang)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

the official test set on TIRA platform [3].

2. Related Work

There is a lot of research in the field of author profiling. Many similar tasks are presented [4, 5, 6, 7] these years, with different focuses such as author gender profiling, bot author detection, fake news spreaders detection, and hate speech spreaders detection. While they focus on different aspects of authors, the general goals are to classify authors on social media, depending on the text content they write. In these tasks, both traditional machine learning models and deep learning models are exploited commonly. Besides, external resources like word embeddings [8] have been widely used to extract features from raw texts. Typical framework for these tasks can be summarized as 1) extract features from the text, which includes word n-grams, TF-IDF features, word embeddings, sentence embeddings, representation from large-scale pretrained language models, and sometimes linguistic features like POS tagging [8, 9, 10, 11]. 2) these features are then fed to classifiers like SVM, BiLSTM, or pretrained language models [12, 13, 14] in order to classify the authors. In these works, there are usually many kinds of features and classifiers selected, and many different combinations of features and classifiers are tested to select the best system. While this framework usually requires arduous work, the resulting system can usually achieve very good performance on the task.

3. Methodology

3.1. Framework

First of all, let X be the tweets of a user, and let y be the label of the user. We intend to predict the label of a user, given his/her tweet data. In other words, we try to estimate the probability distribution $P(y|X)$. Instead of directly building up a model to do this task, we apply Bayes' theorem to the target distribution.

$$P(y|X) \propto P(X|y)P(y) \quad (1)$$

where the target distribution $P(y|X)$ (posterior) is proportional to likelihood $P(X|y)$ times prior $P(y)$. In this way, we convert the desired probability into two components. Once we know $P(X|y)$ and $P(y)$, we will know $P(y|X)$ and thus which label should be assigned to the given tweets.

The prior $P(y)$ is the probability distribution of labels in the dataset, which can be easily obtained by counting the number of a specific label n_i and counting the number of labels in total n , then $P(y_i) = n_i/n$.

The likelihood $P(X|y)$ means that given the label (e.g., the tweets are produced by an irony and stereotype spreader), how likely are these tweets to appear. We can use language models to estimate this probability. The language models are able to predict the probability of any given sequence of words.

During the training stage, we separate the training data according to the labels, more specifically, one collection of data consists of all the ironic (we will use this word as a shorthand for

tweets produced by irony and stereotype spreaders) tweets, another collection consists of all the non-ironic tweets. Namely, $S_I = \{X_i|y_i = \text{"I"}\}$, and $S_{NI} = \{X_i|y_i = \text{"NI"}\}$, where "I" denotes ironic, and "NI" denotes non-ironic. Then we train two separate language models on the two collections of data respectively, which brings about ironic domain language model LM_I and non-ironic domain language model LM_{NI} . Since the language model LM_I is trained on all the ironic tweets, it can predict the probability of a given sequence of words appearing in the ironic domain. In other words, LM_I can estimate the probability $P(X|y = \text{"I"})$, that is, given that the writer is an ironic user, how likely is the sequence to appear. Similarly, LM_{NI} can estimate the probability $P(X|y = \text{"NI"})$. To make it clear, in the training stage, we do not explicitly predict the labels. We only use the labels to divide all the tweets into two collections.

When using our models to predict the label of a given set of tweets, we follow the framework of Bayes' theorem specified above. Firstly, the tweets X are fed into LM_I and LM_{NI} , giving rise to $P(X|y = \text{"I"})$ and $P(X|y = \text{"NI"})$ respectively. Then the likelihood is multiplied by $P(y)$ which can be computed as the frequency of each label in training data, producing $P(X|y = \text{"I"})P(y = \text{"I"})$ and $P(X|y = \text{"NI"})P(y = \text{"NI"})$. By comparing these two, we choose the label that produces the greater posterior as our prediction. Note that in the training data, the labels are evenly distributed. The number of ironic users is equal to the number of non-ironic users, namely $P(y = \text{"I"}) = P(y = \text{"NI"})$. Thus we can further remove the term $P(y)$ and compare only the $P(X|y)$.

For example, given a sentence "I love COVID!", which is pretty ironic, the probability of that occurring in the ironic domain is relatively larger than it occurring in the non-ironic domain. It is not likely for a normal user to say so, while it is likely that an ironic spreader would say some sentences like this. Thus for $X = \text{"I love COVID!"}$, $P(X|y = \text{"I"}) > P(X|y = \text{"NI"})$ if the two language models work effectively. We can then predict that the corresponding label is "I", ironic.

3.2. Language Models

As for the language model, one of our choices is transformer-based neural language models. Specifically, we mainly use the causal language models, GPT2 [15] and DistilGPT2 [16]. Given a sequence of tokens, the models are able to predict the next token based on the previous tokens. In other words, the models can estimate the probability of each token $P(w_i|w_1, w_2, \dots, w_{i-1})$, where $(w_1, w_2, \dots, w_{i-1}, w_i, w_{i+1}, \dots, w_n)$ is a sequence of tokens. The probability of a whole sequence can be obtained as

$$P(w_1, w_2, \dots, w_n) = \prod_{i=1}^n P(w_i|w_1, w_2, \dots, w_{i-1}) \quad (2)$$

In the task IROSTEREO, we need to predict the label for a given set of tweets generated by one user (200 tweets for each user). To estimate the probability of all the tweets $P(\text{tweets})$, we do it in two alternative ways: 1) $P(\text{tweets}) = \prod_{k=1}^N P(\text{tweet}_k)$, where $P(\text{tweet}_k)$ is the probability of a single tweet, which is computed using equation 2. In this way, the tweets are considered independent. 2) Concatenate all the tweets as a single sequence, then divide it into pieces whose length is equal to the maximum input length of the language model. Since the maximum input length is quite long (e.g., 1024 tokens for DistilGPT2), one piece typically

contains multiple tweets. The probability is computed as $P(\text{tweets}) = \prod_{j=1}^N P(\text{piece}_j)$. Thus in this way, the language model is able to attend much longer context on average when estimating the probability for each token. In the following part, we refer to our approach as *BayLMs*, which contains two language models. We refer the approach using 1) to train and predict as *BayLMs_{sep}*, the one using 2) as *BayLMs_{cat}*.

Another kind of important language model we use is the n-gram language model. The main idea of the n-gram language model is that instead of computing the probability of the word (now we use "word" instead of "token" because words are the basic units in this case) based on its whole history (i.e. $P(w_i|w_1, w_2, \dots, w_{i-1})$), it approximates the history with a few words before (i.e., $P(w_i|w_{i-2}, w_{i-1})$ in the case of trigram). In other words, the n-gram language model assumes the probability of a word only depends on its previous n-1 words. Therefore, an n-gram language model estimates the probability of a sequence as

$$P(w_1, w_2, \dots, w_n) = \prod_{i=1}^n P(w_i|w_{i-N+1}, \dots, w_{i-1}) \quad (3)$$

where N denotes N-gram.

To compute the probability of all the tweets written by a single user, we simply concatenate all the tweets as a single sequence, then apply the n-gram language model as 3.

3.3. Further Analysis

In sum, we do not train the models to explicitly do classification. We train two separate language models with two different corpora. With their ability to estimate the probability of any given sequence, the predictions can be derived following Bayes' theorem.

By this approach, we avoid manually selecting and extracting features. Otherwise, due to the long length of input (200 tweets) of each instance, we have to carefully extract features to reduce the length of the input, which is tricky and is possible to lose information. But this framework is only suitable when there are just a few classes ("I" and "NI" in our case), since for each class we have to train a language model.

4. Evaluation

4.1. Data

Since we only have the ground truth included in the training data, we evaluate our approach using training data. Because of the small number of instances in the dataset, we need to evaluate multiple times to get a reliable result and to reduce randomness. Therefore, we use 5-fold cross validation with the training data. More specifically, there are 420 instances in training data with the equal number of ironic and non-ironic instances, thus each fold includes 84 instances. Note that we also keep the ironic and non-ironic instances equal in each fold, thus in each time of training and validation, the number of ironic and non-ironic instances are always equal.

To better compare the performance of the different schemes, we keep the 5 folds always the same. That is, we first randomly divide training data into 5 folds, then we always use these 5 folds to test different schemes and never re-divide again.

Scheme	Accuracy (%)
$BayLMs_{sep}$ with GPT2	91.0 (± 3.6)
$BayLMs_{cat}$ with GPT2	88.6 (± 5.8)
$BayLMs_{sep}$ with DistilGPT2	91.7 (± 4.5)
$BayLMs_{cat}$ with DistilGPT2	91.2 (± 3.6)
$BayLMs$ with unigram LM	88.3 (± 3.5)
$BayLMs$ with bigram LM	90.0 (± 6.8)
$BayLMs$ with trigram LM	91.2 (± 3.6)
$BayLMs$ with 4-gram LM	88.1 (± 4.5)

Table 1

The performance on 5-fold cross validation using training data. The number in each cell of the table is the mean accuracy. The numbers in the brackets are 2 times the standard deviation of the accuracy values in cross validation, which covers a range that most of accuracy values would fall in in theory, assuming the values are normally distributed. Note that due to the small number of instances in training data, there are only 84 instances in each time of validation, thus the accuracy varies wildly.

4.2. Implementation

Our neural language models are implemented using HuggingFace Transformers [17]. They are pretrained on large-scale text corpus beforehand and do causal language modeling during our training process (same task as they did in pretraining stage). They are optimized using stochastic gradient descent with AdamW [18] as the optimizer, with weight decay of 0.01. The language models are trained for just one epoch with a constant learning rate. The batch size and learning rate are adjusted to suit each scheme respectively. The batch size and learning rate for $BayLMs_{sep}$ are 8 and 1e-4. The batch size and learning rate for $BayLMs_{cat}$ are 4 and 2e-5.

As for n-gram language models, we apply add-k smoothing (Laplace smoothing when k=1). We also set the vocabulary of LM_I and LM_{NI} as the union of the vocabulary of ironic tweets and the vocabulary of non-ironic tweets in order to make the comparison between LM_I and LM_{NI} fair when making the prediction. We conduct experiments with unigram, bigram, trigram, and 4-gram language models. For each language model, we select the best k value ranging from 0.01 to 10.0 for add-k smoothing according to the accuracy of cross validation. The selected k for each models are: 0.01 (unigram), 3.0 (bigram), 0.1 (trigram), 0.01 (4-gram).

4.3. Results

The results are presented in table 1. In each cell, the number is the mean accuracy in the cross-validation procedure. The number in the bracket is 2 times the standard deviation of the accuracy values, which covers a range that most of the accuracy values would fall in theoretically, assuming the values are normally distributed. We can see that the accuracy varies wildly in cross validation. This is because of the small number of instances in training data, which results in only 84 instances each time of validation. In the table, GPT2 represents using pretrained GPT2 as the language models, DistilGPT2 means using pretrained DistilGPT2 as language models, and n-gram LM means using n-gram language models.

First of all, we can see that even though the models are not trained under classification tasks, they are able to distinguish between ironic spreaders and normal users effectively. With high

probability, our approach can make accurate predictions.

The results also show that our framework is able to work with entirely different language models, since both *BayLMs* with GPT2/DistilGPT2 and *BayLMs* with n-gram language models achieve good performance.

It is interesting that n-gram language models achieve comparable results with large-scale pretrained language models, even though they are quite simple, do not contain any trainable parameters, and do not require any optimization. They achieve good results simply by counting frequencies in the training data, while neural network-based language models have millions of parameters and require costly optimization.

As for neural network language models, we can see that using DistilGPT2 is slightly better than using GPT2 in general, even though DistilGPT2 has fewer parameters. *BayLMs_{sep}* is slightly better than *BayLMs_{cat}* when they both use GPT2, while *BayLMs_{sep}* and *BayLMs_{cat}* has similar performance when using DistilGPT2. In general, even though neural network language models perform slightly better than n-gram language models, the performance under different settings are similar, which means different settings do not matter significantly. Our final result on the test set of the IROSTEREO task is 92.78%, which is obtained by training *BayLMs_{cat}* with DistilGPT2 on the whole training data.

5. Other Experiments

5.1. Data

There are also some interesting results in our experiments that are noteworthy. In this section, our experiments are conducted using another way to divide training data. We randomly divide the training instances into two sets, the training set which consists of 320 users, and the valid set which consists of 100 users. Again, the ironic and non-ironic instances are equal in both sets. As we mentioned before, we also use the same training and valid sets throughout this section and do not re-divide again. But we do not do cross validation anymore, so the experiments are much less time-consuming. In this setting, the valid set contains only 100 instances and we will only validate once, so the accuracy is not as reliable as the one in table 1. But it is still enough to reflect a change on real accuracy when there is a drastic change occurred.

5.2. The Effect of Pretraining

In our approach, we compare the probability of the given tweets produced by two language models. The prediction of our approach depends on the relative magnitude of probability. $P(X|y = "I")$ and $P(X|y = "NI")$ can both increase or decrease, as long as relative relationship (i.e., $P(X|y = "I") > P(X|y = "NI")$) does not change, the prediction will not change.

In this experiment, we use pretrained and un-pretrained DistilGPT2 as our language models. Un-pretrained DistilGPT2 means that the parameters of the language model are randomly initialized, and the language models are then trained from scratch using our training data. While the pretrained DistilGPT2 language models have the parameters that are sufficiently optimized on a large-scale corpus. We use *BayLMs_{sep}*. As for hyper-parameters, we use batch size of 32, a constant learning rate of 1e-4 and weight decay of 0.01, and train language models

	Accuracy %	Loss of LM_I	Loss of LM_{NI}
$BayLMs_{sep}$ with un-pretrained DistilGPT2	91	5.17	5.66
$BayLMs_{sep}$ with pretrained DistilGPT2	91	3.73	4.00

Table 2

Compare the results of our approach using pretrained and un-pretrained language model. The accuracy is the performance on the valid set. The loss of LM_I is the final training loss of the LM_I on ironic tweets collection. While loss of LM_{NI} is the final training loss of the LM_{NI} on non-ironic tweets collection.

for one epoch. The results are shown in table 2. Note that the accuracy is obtained on the valid set. The loss values are the training loss of the two language models on the two training corpus respectively.

From table 2 we can see that using un-pretrained language model can even achieve the comparable accuracy as using pretrained language model. While pretraining does affect the model performance on language modeling tasks, as the training loss is much larger when the model is not pretrained, pretraining does not affect the performance of our $BayLMs$ to a significant extent. This unexpected result is because of the intuition we mentioned above. Without pretraining, the language models are worse at estimating the probability of a given sequence, in other words, fail to assign a high probability to the occurring sequence. Without pretraining, both $P(X|y = "I")$ and $P(X|y = "NI")$ are likely to decrease, while the relative relationship is not likely to change.

Note that in other parts of this paper, we use pretrained language models in our $BayLMs$.

5.3. Masked Language Models

It is also possible to use masked language models in $BayLMs$, (e.g. BERT [14]). Even though we cannot estimate strictly the probability of the whole tweets, we can do it in an analogous way to causal language models. During training, the masked language models are trained to predict the masked token in the sequence. When we use them in $BayLMs$ to make predictions, we randomly mask tokens in the input tweets, then the identical masked inputs are fed into the two masked language models, giving rise to probabilities on the masked tokens, which are then multiplied together to produce $P(X|y = "I")$ and $P(X|y = "NI")$.

Specifically, we use pretrained DistilRoBERTa-base [16] and BERTweet-base [19]. The latter is pretrained on a large-scale tweet corpus, and has special tokens for emojis, user mentions, and url links. Thus it has a strong ability to understand tweets. We also use $BayLMs_{sep}$ as framework. As for hyper-parameters, both DistilRoBERTa and BERTweet are trained with batch size=32, constant learning rate=2e-5, weight decay=0.01 for one epoch. The masking rate for random masking on inputs is 0.15 during training and evaluation. The results are shown in table 3.

We can see that the DistilRoBERTa and BERTweet give comparable accuracy. But the performance of both models is noticeably worse than $BayLMs$ with causal language models. It may be because they fail to estimate the probability of all the tokens and finally the probability of the whole input tweets.

	Accuracy
<i>BayLMs_{sep}</i> with DistilRoBERTa-base	76
<i>BayLMs_{sep}</i> with BERTweet-base	79

Table 3

The results when using masked language models in our approach. The accuracy is the performance on valid set.

	Accuracy %	Loss of LM_I	Loss of LM_{NI}
<i>BayLMs_{sep}</i> trained for 0.5 epoch	87	3.80	4.11
<i>BayLMs_{sep}</i> trained for 1 epoch	92	3.79	4.03
<i>BayLMs_{sep}</i> trained for 3 epoch	86	3.24	3.45
<i>BayLMs_{sep}</i> trained for 6 epoch	61	2.63	2.82

Table 4

Compare the results of our approach with different training epochs. The accuracy is the performance on the valid set. The loss of LM_I is the final training loss of the LM_I on ironic tweets collection. While loss of LM_{NI} is the final training loss of the LM_{NI} on non-ironic tweets collection.

5.4. Effect of Training Epochs

In most cases, training only for one epoch is considered to be insufficient. But our approach is different from the typical method, the situation can be a little bit different. We use the *BayLMs_{sep}* with pretrained DistilGPT2, and train language models with exactly the same hyper-parameters (batch size=8, constant learning rate=1e-4, weight decay=0.01) but for the different number of epochs, ranging from 1 to 6. The results are shown in table 4.

As we can see, when the number of epochs exceeds 1, the accuracy starts to drop. *BayLMs_{sep}* trained for 6 epochs performs significantly worse than *BayLMs_{sep}* trained for 1 epoch. Typically, machine learning models are trained for much more epochs to achieve the best results, and models' performance starts to decline only after training for many epochs, and it usually declines mildly. While the decline in other machine learning models can be attributed to over-fitting, the reason for the decline in our case can not be explained as over-fitting, since the models do not do the same task in the evaluation as in training. In our approach, it is necessary to train the model, since the $P(X|y = "I")$ and $P(X|y = "NI")$ will always be the same if the language models are not trained. But the accuracy will not necessarily increase with more training. Like we mentioned in section 5.2, the prediction accuracy depends on the relative relationship of $P(X|y = "I")$ and $P(X|y = "NI")$, instead of the absolute magnitude of probabilities. It is also noteworthy that our system has a strong bias towards the class "NI" when trained for too many epochs. In the case of 6 epochs, there are 38 out of 39 wrong predictions are "NI" (corresponding ground truths are "I"). While in the case of 1 epoch, there are 5 wrong predictions are "NI" and 3 are "I".

6. Conclusion and Future Work

This paper has shown a novel approach when dealing with classification tasks. We present a simple scheme, which only makes use of language models and trains with the language modeling

task. The proposed system is simple to implement while it can achieve good performance on the task.

In theory, our approach is compatible with any kind of language model, as long as the language model is able to give the probability of a given sequence. This point is well demonstrated by our experimental results. Thus, we can further test with other language models in future work, including different types of neural network language models and other traditional ones. In addition, regarding n-gram language models there are some sophisticated smoothing methods that are usually believed to achieve better results than simple add-k smoothing. So we can also do some experiments with them.

References

- [1] O.-B. Reynier, C. Berta, R. Francisco, R. Paolo, F. Elisabetta, Profiling Irony and Stereotype Spreaders on Twitter (IROSTEREO) at PAN 2022, in: CLEF 2022 Labs and Workshops, Notebook Papers, CEUR-WS.org, 2022.
- [2] J. Bevendorff, B. Chulvi, E. Fersini, A. Heini, M. Kestemont, K. Kredens, M. Mayerl, R. Ortega-Bueno, P. Pezik, M. Potthast, F. Rangel, P. Rosso, E. Stamatatos, B. Stein, M. Wiegmann, M. Wolska, E. Zangerle, Overview of PAN 2022: Authorship Verification, Profiling Irony and Stereotype Spreaders, and Style Change Detection, in: M. D. E. F. S. C. M. G. P. A. H. M. P. G. F. N. F. Alberto Barron-Cedeno, Giovanni Da San Martino (Ed.), *Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Thirteenth International Conference of the CLEF Association (CLEF 2022)*, volume 13390 of *Lecture Notes in Computer Science*, Springer, 2022.
- [3] M. Potthast, T. Gollub, M. Wiegmann, B. Stein, TIRA Integrated Research Architecture, in: N. Ferro, C. Peters (Eds.), *Information Retrieval Evaluation in a Changing World, The Information Retrieval Series*, Springer, Berlin Heidelberg New York, 2019. doi:10.1007/978-3-030-22948-1_5.
- [4] F. Rangel, P. Rosso, M. Montes-y Gómez, M. Potthast, B. Stein, Overview of the 6th author profiling task at pan 2018: multimodal gender identification in twitter, *Working Notes Papers of the CLEF (2018)* 1–38.
- [5] F. Rangel, P. Rosso, Overview of the 7th author profiling task at pan 2019: bots and gender profiling in twitter, in: *Working Notes Papers of the CLEF 2019 Evaluation Labs Volume 2380 of CEUR Workshop*, 2019.
- [6] F. Rangel, A. Giachanou, B. H. H. Ghanem, P. Rosso, Overview of the 8th author profiling task at pan 2020: Profiling fake news spreaders on twitter, in: *CEUR Workshop Proceedings*, volume 2696, Sun SITE Central Europe, 2020, pp. 1–18.
- [7] F. Rangel, G. L. De la Peña Sarracén, B. Chulvi, E. Fersini, P. Rosso, Profiling hate speech spreaders on twitter task at pan 2021., in: *CLEF (Working Notes)*, 2021, pp. 1772–1789.
- [8] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, 2013. URL: <https://arxiv.org/abs/1301.3781>. doi:10.48550/ARXIV.1301.3781.
- [9] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in:

- Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.
- [10] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, et al., Universal sentence encoder, arXiv preprint arXiv:1803.11175 (2018).
 - [11] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer, Deep contextualized word representations, 2018. URL: <https://arxiv.org/abs/1802.05365>. doi:10.48550/ARXIV.1802.05365.
 - [12] C. Cortes, V. Vapnik, Support-vector networks, *Machine learning* 20 (1995) 273–297.
 - [13] S. Hochreiter, J. Schmidhuber, Long short-term memory, *Neural computation* 9 (1997) 1735–1780.
 - [14] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).
 - [15] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., Language models are unsupervised multitask learners, *OpenAI blog* 1 (2019) 9.
 - [16] V. Sanh, L. Debut, J. Chaumond, T. Wolf, Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2019. URL: <https://arxiv.org/abs/1910.01108>. doi:10.48550/ARXIV.1910.01108.
 - [17] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, A. M. Rush, Huggingface’s transformers: State-of-the-art natural language processing, 2019. URL: <https://arxiv.org/abs/1910.03771>. doi:10.48550/ARXIV.1910.03771.
 - [18] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, 2017. URL: <https://arxiv.org/abs/1711.05101>. doi:10.48550/ARXIV.1711.05101.
 - [19] D. Q. Nguyen, T. Vu, A. T. Nguyen, BERTweet: A pre-trained language model for English Tweets, in: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020, pp. 9–14.