Computational Intelligence Lab - Text Classification

Philipp Guldimann, Alec Dorrington, Leon Windler
Group: PALs
Department of Computer Science, ETH Zürich, Switzerland

Abstract— **Text sentiment analysis involves the application of natural language processing and machine learning to identify the sentiment or emotional tone in a piece of text. We apply sentiment analysis techniques to determine whether *tweets* from Twitter are of positive or negative tone. A number of techniques are explored, including *bag of words* and *neural networks*. Our aim is to identify, among those techniques considered, some with promising efficacy in this area. Furthermore, we will attempt to effectively combine these base techniques to establish a new mixed model.**

I. Motivation and Background

Text sentiment analysis, also known as opinion mining or sentiment analysis, is the process of determining and categorizing the sentiment or emotional tone expressed in a piece of text, typically through the use of natural language processing (NLP) and machine learning techniques.

The goal of sentiment analysis is to understand the subjective attitudes, opinions, and emotions conveyed in text data, such as customer reviews, social media posts, news articles, or any other textual content. This can be a valuable tool when it comes to customer feedback, market research, or political analysis.

One website that lends itself well to sentiment analysis is the micro-blogging service Twitter. Here, millions of daily users come together to discuss the latest news, products, events, and more. Hence, by filtering and processing their tweets, we can quickly gain valuable insights into public sentiment, customer opinions, and market trends, enabling organizations and individuals to make informed decisions, improve customer experiences, and respond effectively to various local and global situations.

II. Methodology

Here, we compare and contrast an assortment of methods for analysing the sentiment of individual tweets, in the order that we may discover which approaches work the best. Our analysis is limited to determining whether the overall tone of a tweet is positive or negative. We make no attempt to make more fine-grained categorisations nor do we assign different categorisations to different potions of a tweet.

In the final step, we attempt to combine our best models into a single mixed model, where a judge model is assigned the task of identifying which model is most likely to produce a correct result in each instance.

A. Dataset

All considered models are trained on the same dataset, a collection of labelled tweets provided by ETH Zürich with 1,250,000 positive examples and 1,250,000 negative examples. This data is preprocessed, with usernames and URLs replaced by placeholders.

B. Evaluation

Comparisons are made on the basis of the overall classification accuracy, this being the portion of tweets in the test set to have their sentiment correctly determined by each model. Note that the dataset is balanced as there exists an equal portion of positive and negative examples. For the purpose of determining accuracy, 5-fold cross-validation is applied with 5% of the dataset being held out for testing in each case.

Fig. 1: The validation accuracy as determined by the number of samples N, the true labels Y, and the predicted labels $\tilde{Y}$, where $Y \in [-1, 1]^N$.

$$Accuracy = \frac{1}{N} \sum_{i \in [N]} \mathbb{1}_{sign(\tilde{y_i}) = y_i}$$

III. Models

Implementations for all models discussed can be found at `https://github.com/lonlon11/cil2023`.

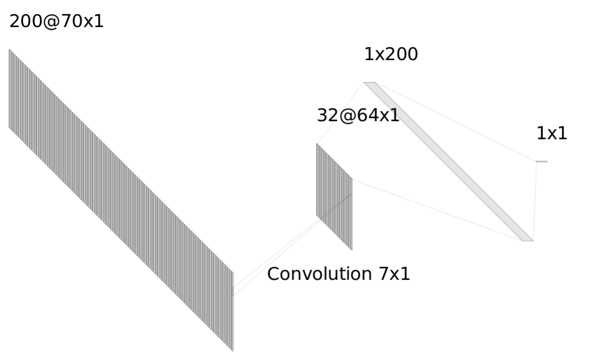
A. Bag of Words

The simplest model considered is the linear *bag of words* model. Disregarding any inherent structure of the documents e.g. grammar, word order, or context, it represents them as a collection of their individual words. Hence, each tweet is a vector of counts in which each element corresponds to a word from the dictionary. A linear model with hinge loss is fitted to the word count vector using stochastic gradient descent in *scikit-learn*. Here, we can learn the typical sentiment and strength of individual words, but we have no way of learning higher-order relationships *between* different words

B. Convolutional Neural Network

Next, we apply a 1-dimensional *convolutional neural network* [7] (CNN) model, implemented using *PyTorch*. In general, due to their increased modelling power, neural networks are able to capture more general inter-word dependencies. If we make the further assumption that such dependencies exhibit some form of shift-invariance, a convolutional layer can be used to find patterns in a sliding window.

Fig. 2: The architecture of our CNN model.

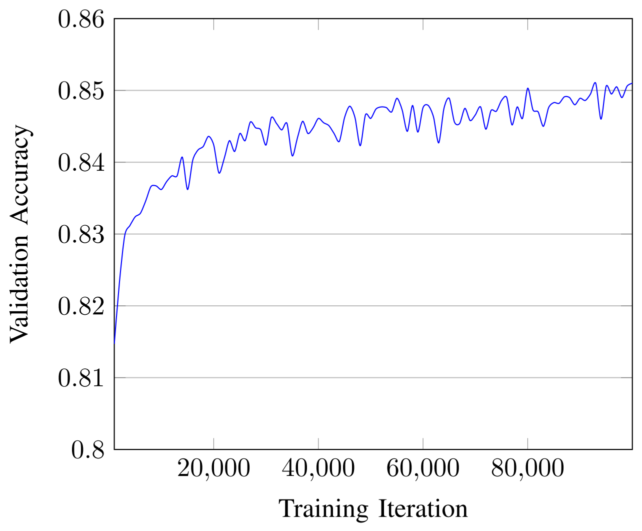


All words in the sentence are embedded into a 200-dimensional latent space using GloVe encodings [9] from Stanford. Hopefully, similar words should have similar encodings.

Our CNN uses a single convolutional layer, which slides over all contiguous 7-tuples of word embeddings and produces 32 filters. Next, this is flattened and fed into a single 100-dimensional fully-connected layer, which is connected to the single output. The use of additional convolutional or fully-connected layers did not seem to confer any advantage. All layers except the last use a *leaky-ReLU* activation, while the final layer uses *tanh*, since its range matches the required output format. The network was trained with ADAM for $200,000$ iterations with a batch size of 50.

We also tried filtering out *stop-words* such as 'the' and 'a', as is standard practice, since these words are often considered irrelevant for any analysis purposes. However, this lead to significantly degraded performance.

Fig. 3: The evolution of the validation accuracy for our CNN model as training progresses, sampled once after every $1,000$ iterations.



C. Transformer Models

Transformer models are able to update the latent representations of words based on a learned *attention* mechanism which finds related words in a limited context window [13], thus better identifying the meaning of a word in its current context.

A 2018 paper [5] by Mariel et al influenced our decision to try a BERT model for this task, a kind of transformer model, although their model was primarily focused on Indonesian text. Nowadays, transformer models are the standard approach taken when dealing with textual data.

A transformers consists of an encoder and a decoder part. For a classification task such as this, it is common to only use an encoder. Using transformers in such a way is often described as *discriminative* learning. Use of a corresponding decoder is only necessary for generative tasks.

Transformer models famously require a large amount of time and data to train. However, they are also often quite amenable to fine-tuning. As such, for our transformer model implementation, we have elected to start with two pre-trained transformer models and fine-tune them.

The two base models we tried are the following:

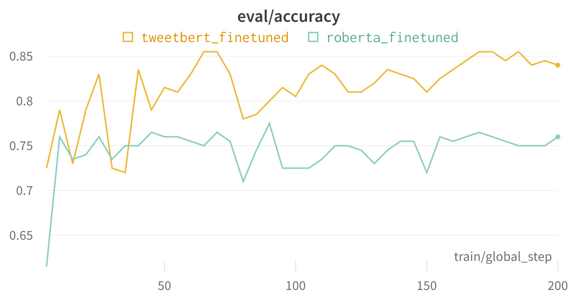
- SiEBERT [2], which itself is a fine-tuned version of RoBERTa-large.
- A model from [10], a fine-tuned version of BERTweet [6] specifically for sentiment analysis.

Both of the preceding models are available on HuggingFace[1].

Each of these transformer models was fine-tuned using our tweet sentiment data, although the base models are also tested for the sake of comparison. For the purpose of fine-tuning, a GTX 1080 Ti GPU on the ETH Euler cluster was used. The effective batch size was 8, simulated with an actual batch size of 1 with accumulated gradient steps in order that the training data may fit in memory.

After every 5 batches of training, an evaluation is performed on the separate evaluation set, so that we may see how the performance is effected by further training. In both cases, the accuracy improves sharply at the beginning and very quickly starts to level off after around 50 iterations, then offering little further improvement.

Fig. 4: The evolution of the validation accuracy for our fine-tuned BERT models as training progresses.



[1]https://huggingface.co/

D. Judge Model

Our two best models up until this point are the CNN and the fine-tuned BERTweet model, each with about 85% accuracy.

Empirically, the probability that *both* models get a correct answer is approximately 0.775, and the probability that *at least one* model gets a correct answer is 0.925, and both models agree (rightly or wrongly) 0.851 of the time.

Say we want to train an additional *judge* model whose job it is to predict which of the two above models is more likely to get the correct answer, and then output the response from that model. Assuming such a judge were capable of choosing a model perfectly gives us a theoretical upper bound on its accuracy of approximately 0.925. However, such a result is unlikely, due in part to the fact that some of the correct results from each model are inevitably due to luck (indeed, we might assume that around half of the samples that each model doesn't know how to classify are nonetheless categorised correctly).

Similar to the *bag of words* model, we use the $CountVectorizer$ from *scikit-learn* to create embeddings for tweets based on word frequencies, and then fit a linear classifier. For our training data, we consider all cases where the two base models disagree, and then expect the classifier to predict, based on the word frequencies, which model was correct.

Fig. 5: The performance of the judge model.

Validation Accuracy		
Model version	Judge component	Combined model
Small judge	0.480	0.660
Big judge	0.716	0.846

In the above, *small judge* refers to a judge trained on just $10,000$ samples, and *large judge* is one trained on $199,000$ samples. The accuracy of the judge component is how often the judge selected the correct model, and the accuracy of the combined model is how well the judge model performed overall in the original sentiment analysis task.

The judge is trained on different training samples than the base models, but the base models are trained on the same samples as each other. Clearly, the number of training examples offered to the judge has a substantial impact on the performance. However, the use of a judge was unfortunately unable to improve the performance. Perhaps a more sophisticated judge would have fared better.

IV. Other Models

Naturally, we were not able to try every method. Here, we outline some of the available alternatives that we did not consider.

A. Lexicon-based approach

Given a lexicon of words or phrases, and a positive/negative label for each phrase, one can search for occurrences of each phrase in the text and sum up the sentiment scores from found phrases. Comparing this score to a threshold gives rise to the classification. This method is largely superseded by the CNN model, which can essentially learn a lexicon automatically from given input data.

B. Support Vector Machine

A classic supervised learning algorithm is the *support vector machine* (*SVM*). Here, the *SVM* tries to find the optimal decision boundary that separates the samples of different sentiment classes in the feature space. The tweet samples which are closest to that boundary are the support vectors. The maximisation of the margin between this decision boundary and the support vectors leads to good generalisation performance. As they are able to handle non-linear decision boundaries, they are capable of capturing complex patterns and relationships in their feature space and thus perhaps recognise difficult sentiment patterns.

C. Naive Bayes

Another possible approach is Naive Bayes (*NB*). Here, we assume that the words in the text are conditionally independent given the sentiment category. This simplifying assumption greatly reduces the computational complexity and allows for efficient training and classification. One downside can be that due to this independence assumption, an NB model can't capture the intrinsic dependencies between words.

D. Topic Modelling

Topic modelling is an approach often applied in classical document analysis. The idea, similar to matrix decomposition, is to find a latent representation for each document which reveals the presence of diverse topics and their respective frequencies. Each topic is characterized by a set of the most frequent words associated with it.

In our case, we can look at topic modelling as a generalisation of *bag of words*, where bag of words corresponds to a rank-1 version of a topic model. The use of more topics would allow one to better handle words which change meaning depending on their context (surrounding words). However, the CNN and transformer models also grant this capability, and the breakdown of sentiment by topic is outside the scope of this research, perhaps making this extra complexity unnecessary.

V. Results

Fig. 6: The final validation accuracy for each of our models.

Model	Accuracy
Bag of Words	0.801
Convolutional Neural Network	0.853
Judge	0.846
RoBERTa Base	0.663
RoBERTa Fine-tuned	0.750
BERTweet Base	0.696
BERTweet Fine-tuned	0.848

All models managed to perform better than guessing randomly (0.500).

The clear victors here are the *CNN* and *BERTweet Fine-tuned* models, which performed about as well as each other.

An honorable mention should be awarded to the linear *bag of words* model, which despite its simplicity, was able to achieve results not far behind the two front-runners, and ahead of 3 of the 4 transformer models, all despite the fact that it was implemented and trained in a fraction of the time than that of the other models.

Perhaps it should not be particularly surprising that the elaborate neural network models won out over bag of words, if only by 5%. What might be more interesting however, is how the transformer model seemed to offer no substantial advantage over the CNN, nor did the use of a pre-trained network. This additional sophistication seems to be unnecessary, as the simpler CNN was able to capture all the complexity that the transformer could.

Among the transformers, both of the fine-tuned models performed better than their respective base versions. This is to be expected, especially in the case of the RoBERTa model which was only ever trained on general English text rather than tweets specifically. However, what may be surprising is that the advantage gained from fine-tuning was greater for the BERTweet model, which unlike RoBERTa was already trained on tweets. Perhaps the total variance in possible tweet strings is greater than what is captured by their, or our, data.

VI. Discussion

A. Limitations

Firstly, it is important to note that this work should not be taken to represent a definitive ranking of the discussed methods. The results attributed to each method are of course limited by our implementations. And while reasonable efforts were made to optimise each model, this by no means guarantees perfection.

Secondly, our models have only been trained and evaluated on tweet data. Twitter, like any large platform, has its own set of social and linguistic conventions. No effort was made to test generalisation to other kinds of text. We also can not comment on how well the ETH-provided dataset represents Twitter as a whole.

Finally, as has been previously mentioned, we only assign a binary sentiment label to text at the level of the entire tweet. Other methods exist which offer more specific sentiment categories and can ascribe separate sentiments on the level of particular topics within a text.

B. Improvements

Based on how quickly the transformer models converged, it seems unlikely that the provision of additional data would help, perhaps unless it were in some way more varied (although our test set is drawn from the same distribution). Perhaps the CNN would benefit more, since it continues to improve until almost all training examples have been seen. However, it seems more likely to us that the limiting factor has to do with the models themselves.

In further research, one could investigate the use of alternative network topologies or activation functions for each of our models, to see if this could offer any improvement. Reasonable attempts were made to explore different hyper-parameters, albeit not methodically. A further improvement in methodology would involve the use of cross-validation to test different hyper-parameters.

Finally, we believe that our combined model approach could be refined to give better results. Such refinements include a different model for the judge component, an improved embedding technique, and potentially a more fine-grained integration between the judge and the base models.

VII. Conclusion

Our findings highlight the strength of CNN and BERTweet Fine-tuned models in sentiment analysis of English tweets. Our preliminary findings lead us to recommend the CNN model for this task, based on its relative simplicity and efficacy.

While our initial attempt to combine a CNN and transformer model into the Judge Model did not yield the desired results, we firmly believe that the integration of these two powerful techniques can lead to significant improvements if executed correctly. Hence, further research into such hybrid approaches could be worthwhile.

References

[1] Rushlene Kaur Bakshi, Navneet Kaur, Ravneet Kaur, and Gurpreet Kaur. Opinion mining and sentiment analysis. *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 452–455, 2016.

[2] Jochen Hartmann, Mark Heitmann, Christian Siebert, and Christina Schamp. More than a feeling: Accuracy and application of sentiment analysis. *International Journal of Research in Marketing*, 40(1):75–87, 2023.

[3] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[4] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[5] Wahyu Calvin Frans Mariel, Siti Mariyah, and Setia Pramana. Sentiment analysis: a comparison of deep learning neural network algorithm with svm and nave bayes for indonesian text. *Journal of Physics: Conference Series*, 971(1):012049, mar 2018.

[6] Dat Quoc Nguyen, Thanh Vu, and Anh Tuan 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*, pages 9–14, 2020.

[7] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *ArXiv e-prints*, 2015.

[8] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1–2):1–135, jan 2008.

[9] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[10] Juan Manuel Pérez, Juan Carlos Giudici, and Franco Luque. pysentimiento: A python toolkit for sentiment analysis and socialnlp tasks, 2021.

[11] Rahmat Syahputra, Gomal Juni Yanris, and Deci Irmayani. Svm and naïve bayes algorithm comparison for user sentiment analysis on twitter. *Sinkron*, 2022.

[12] Harsh Thakkar and Dhiren R. Patel. Approaches for sentiment analysis on twitter: A state-of-art study. *ArXiv*, abs/1512.01043, 2015.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[14] Yasmen Wahba, Nazim Madhavji, and John Steinbacher. A comparison of svm against pre-trained language models (plms) for text classification tasks, 2022.

[15] Lei Zhang and Bing Liu. *Sentiment Analysis and Opinion Mining*, pages 1152–1161. Springer US, Boston, MA, 2017.

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):

Computational Intelligence Lab - Text Classification

Authored by (in block letters):

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

Name(s):	**First name(s):**
Guldimann	Philipp
Dorrington	Alec
Windler	Leon

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, 31.07.2023	PhilippG
	I am Alec
	[illegible]

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