目录

FinBERT: Financial Sentiment Analysis with Pre-trained Language Models

Current efforts:

1. Machine learning methods with features **extracted from text with "word counting"**; inability to represent the semantic information that results from a particular sequence of words
2. Deep learning methods, where text is **represented by a sequence of embeddings**; often deemed as too "data-hungry" as it learns a much higher number of parameters

The purpose behind financial sentiment analysis is usually guessing how the markets will react with the information presented in the text

First papers:

apply an LSTM neural network to ad-hoc company announce- ments to predict stock-market movements and show that method to be more accurate than traditional machine learning approaches.

shortcomings

Due to lack of **large labeled financial datasets**, it is difficult to utilize neural networks to their full potential for sentiment analysis.

ELMo (Embeddings from Language Models)

Using the pre-trained weights of ELMo, contextualized word embeddings can be calculated for any piece of text.

The goal of a language model is to learn the probability distribution over sequences of tokens in a given vocabulary.

ULMFit

Since a text is a sequence of tokens, the first choice for any LSTM natural language processing model is determining how to initially represent a single token.

GLoVe

One such pre-training algorithm is GLoVe (Global Vectors for Word Representation)

Transformer

BERT for financial domain: FinBERT

Further pre-training.

in order to observe if such adaptation is going to be beneficial for financial domain.

1. The first is pre-training the model on a relatively large corpus(语料库) from the target domain.
2. The second approach is pre-training the model only on the sentences from the training classification dataset.

FinBERT for text classification

Sentiment classification is conducted by adding a dense layer after the last hidden state of the [CLS] token.

Then, the classifier network is trained on the labeled sentiment dataset.

FinBERT for regression.

implement regression with almost the same architecture on a different dataset with continuous targets. loss function being used is mean squared error

(used to predict continuous outcomes based on the input text.)

Training strategies to prevent catastrophic forgetting.

1. slanted triangular learning rates: learning rate first linearly increases up to some point and after that point linearly decreases.
2. discriminative fine-tuning : Discriminative fine-tuning is using lower learning rates for lower layers on the network. eg: at layer l , the lr is α; at layer l-1, the lr is $\alpha_{l-1} = \theta \alpha_l$ where the θ should be less than 1;
3. gradual unfreezing. start training with all layers but the classifier layer as frozen. During training we gradually unfreeze all of the layers starting from the **highest one, so that the lower level features become the least fine-tuned ones.**

Experiment:

Datasets:TRC2-financial. Financial PhraseBank;

further pre-train :TRC2-financial main sentiment analysis: Financial PhraseBank the data for Task 1：FiQA Sentiment.

base method:

LSTM classifier with GLoVe embeddings, LSTM classifier with ELMo embeddings, ULMFit classifier.

best word

It should be noted that these baseline methods are not experimented with as thoroughly as we did with BERT. Therefore the results should not be interpreted as definitive conclusions of one method being better. 应该指出的是，这些基线方法并没有像我们对 BERT 所做的那样进行彻底的实验。 因此，结果不应被解释为某种方法更好的明确结论。

Evaluation Metrics

Accuracy, cross entropy loss macro F1 average. Calculates F1 scores for each of the classes and then takes the average of them.

Implementation details

dropout probability of p = 0.1, warm-up proportion of 0.2, maximum sequence length of 64 tokens, a learning rate of 2e – 5 a mini-batch size of 64. train the model for 6 epochs, evaluate on the validation set and choose the best one. set the discrimination rate as 0.85. start training with only the classification layer unfrozen, after each third of a training epoch we unfreeze the next layer.

An Amazon p2.xlarge EC2 instance with one NVIDIA K80 GPU, 4 vCPUs and 64 GiB of host memory is used to train the models.

Result

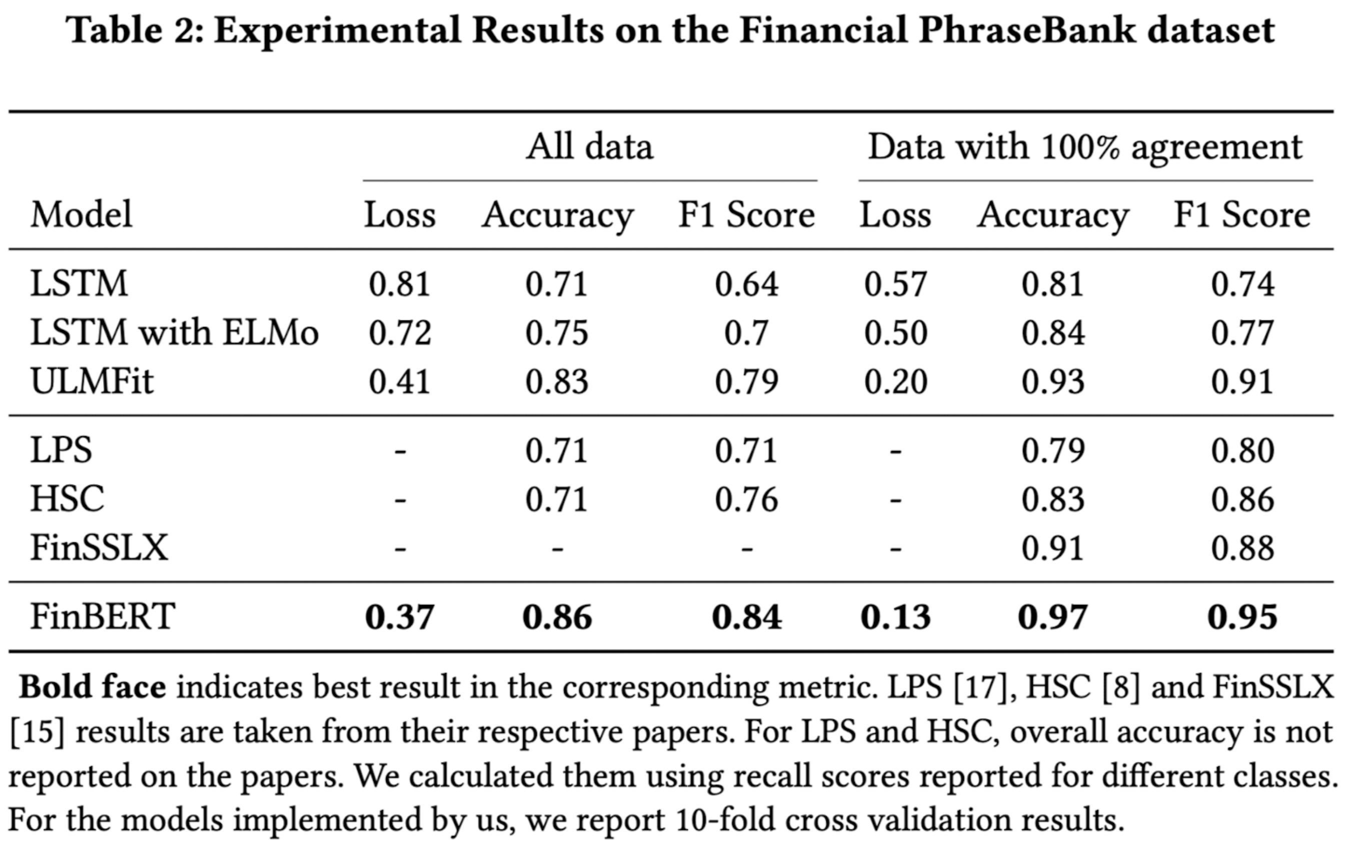
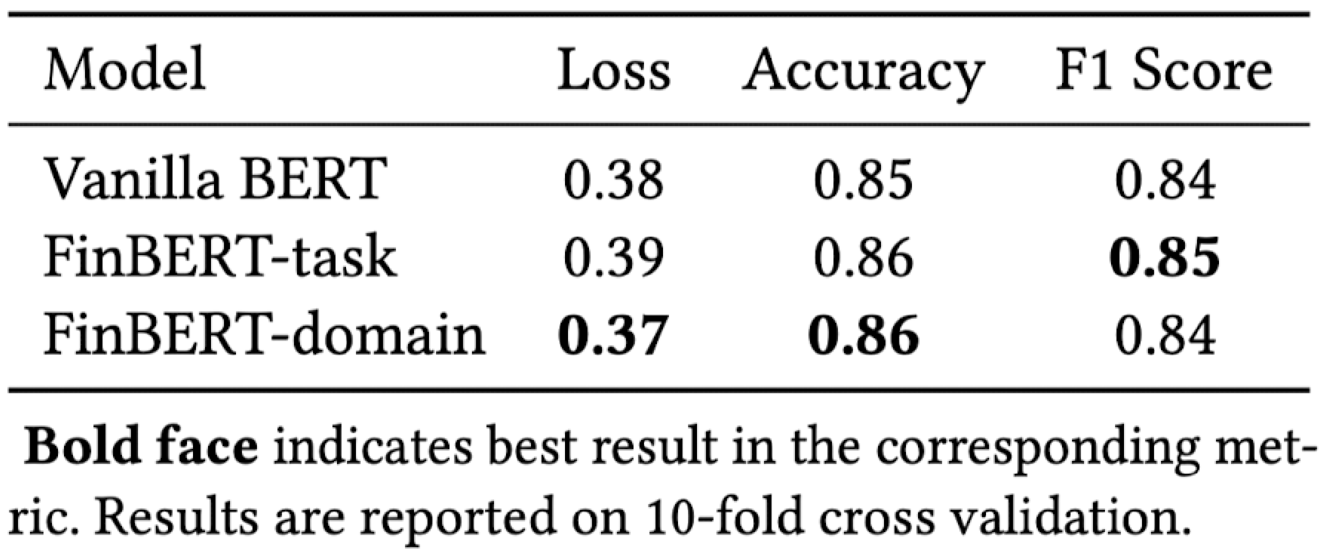


Table 2: Experimental Results on the Financial PhraseBank dataset

	All data			Data with 100% agreement		
Model	Loss	Accuracy	F1 Score	Loss	Accuracy	F1 Score
LSTM	0.81	0.71	0.64	0.57	0.81	0.74
LSTM with ELMo	0.72	0.75	0.7	0.50	0.84	0.77
ULMFit	0.41	0.83	0.79	0.20	0.93	0.91
LPS	-	0.71	0.71	-	0.79	0.80
HSC	-	0.71	0.76	-	0.83	0.86
FinSSLX	-	-	-	-	0.91	0.88
FinBERT	**0.37**	**0.86**	**0.84**	**0.13**	**0.97**	**0.95**

Bold face indicates best result in the corresponding metric. LPS [17], HSC [8] and FinSSLX [15] results are taken from their respective papers. For LPS and HSC, overall accuracy is not reported on the papers. We calculated them using recall scores reported for different classes. For the models implemented by us, we report 10-fold cross validation results.

Do we need further pre-train?



Model	Loss	Accuracy	F1 Score
Vanilla BERT	0.38	0.85	0.84
FinBERT-task	0.39	0.86	**0.85**
FinBERT-domain	**0.37**	**0.86**	0.84

Bold face indicates best result in the corresponding metric. Results are reported on 10-fold cross validation.

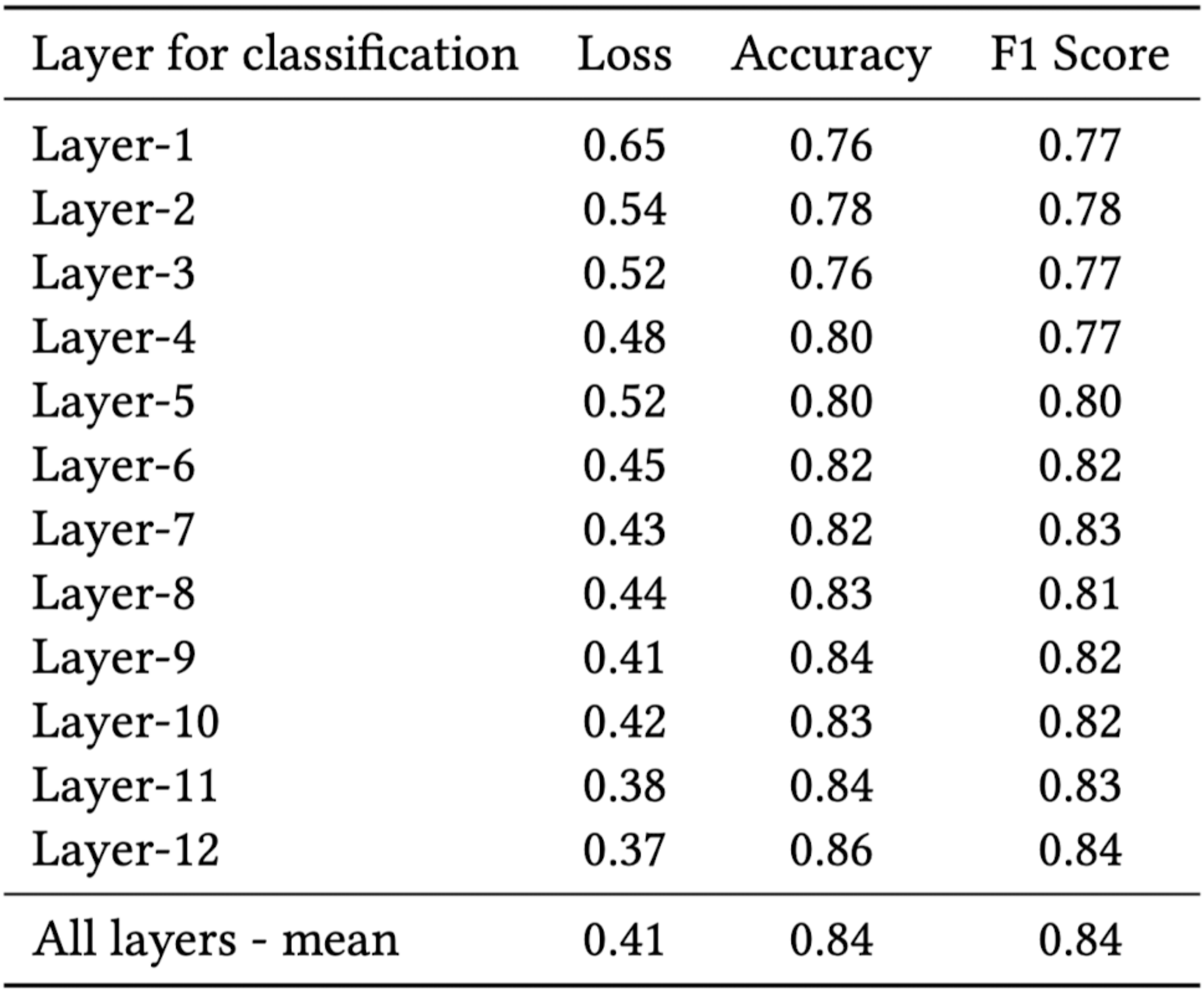
No obvious improvement.

Do we need to consider catastrophic forgetting

It's not important in our project, but it is a shining point of our project.

One way that catastrophic forgetting can show itself is the sud- den increase in validation loss after several epochs.

The best layer for classification



Layer for classification	Loss	Accuracy	F1 Score
Layer-1	0.65	0.76	0.77
Layer-2	0.54	0.78	0.78
Layer-3	0.52	0.76	0.77
Layer-4	0.48	0.80	0.77
Layer-5	0.52	0.80	0.80
Layer-6	0.45	0.82	0.82
Layer-7	0.43	0.82	0.83
Layer-8	0.44	0.83	0.81
Layer-9	0.41	0.84	0.82
Layer-10	0.42	0.83	0.82
Layer-11	0.38	0.84	0.83
Layer-12	0.37	0.86	0.84
All layers - mean	0.41	0.84	0.84

6-th layer gives a good result. 这可能表明两个因素：1）当使用较高层时，正在训练的模型更大，因此可能更强大，2）较低层捕获更深层次的语义信息，因此它们很难微调该信息分类。

Given Idea: Training only a subset of the layers

结果如表7所示。仅微调分类层并不能达到与微调其他层相近的性能。 然而，仅对最后一层进行微调就可以轻松超越 HSC 等最先进的机器学习方法。 在第 9 层之后，性能变得几乎相同，只是通过微调整个模型来超越。 这个结果表明，为了利用 BERT，整个模型的昂贵训练并不是强制性的。 可以在训练时间大大减少和模型性能小幅下降的情况下做出公平的权衡。

Result:

1. 我们的理论是，BERT 在我们的数据集上已经表现得足够好，进一步的预训练没有太大的改进空间。
2. 我们还发现，对较高层进行更积极调整的学习率机制比对较低层进行更积极的调整表现更好，并且在防止灾难性遗忘方面更有效。
3. 我们实验的另一个结论是，通过仅微调 BERT 的最后 2 层，可以用更少的训练时间实现类似的性能。

FinBERT is good enough for extracting explicit sentiments, but modeling implicit information that is not neces- sarily apparent even to those who are writing the text should be a challenging task. Another possible extension can be using FinBERT for other natural language processing tasks such as named entity recognition or question answering in financial domain.

Sentiment classification with adversarial learning and attention mechanism

Traditional classifiers suffer from overfitting problem。

we also aim to solve the positive and unlabeled learning (PU learning) problem.

PU learning： positive labeled and unlabeled reviews, without any negative labeled reviews.

For example, in some e-commerce websites, the polarities of many reviews are not labeled manually or cannot be labeled automatically as there are no associated ratings.

The evaluation metric for PUlearning problem is usually F1-score,precisionorrecall,which cannot be modeled in the current loss function of adversarial learning, which increases the difficulty in optimization during training process.

Keyword: PU learning problems, traditional sentiment classification problem;

Build an attention-based long short-term memory (LSTM) network, enhanced with an improved adversarial learning method.

adversarial learning method:

Adversarial learning is a technique primarily used in the field of machine learning and artificial intelligence where models are trained to become more robust by learning to handle adversarial examples. These adversarial examples are inputs to the model that have been intentionally perturbed or modified in a way that causes the model to make a mistake in its predictions or classifications, despite the modifications often being imperceptible or seemingly irrelevant to humans.

The concept is most famously applied in the context of Generative Adversarial Networks (GANs), where two neural networks are trained simultaneously in a game-theoretic scenario. One network, the generator, learns to generate data (such as images) that is indistinguishable from real data, while the other network, the discriminator, learns to distinguish between real and generated data. The process is **adversarial** because the generator is trying to "fool" the discriminator, and in doing so, it improves its ability to generate data that is similar to the real data. The discriminator, on the other hand, improves its ability to detect the fake data. This process leads to the generation of high-quality data as the networks iteratively improve through this competition.

In the broader context, adversarial learning techniques are used to improve the security and robustness of machine learning models against adversarial attacks, enhance the performance of models in various tasks, and explore the limits of model generalization and interpretation.

word embedding

As a technique of feature extraction from words, word embedding represents a word as a vector containing continuous real numbers. The vector is learned from the context in which words appear in the text, and is used to represent the meaning of the word in that context.

The word embedding results can be represented by a matrix W.

Adversarial learning

we build a Θ-parameterized classifier, where y_n will be predicted by $p(y_n| T_n ; \Theta)$

loss function is

$L(\bar x;\Theta) = - \sum_{n=1}^{N} \log p(y_n| T_n ; \Theta)$

proposed method:

1. $L(\bar x ; \Theta) = L(\bar x + \bar e_{adv} ; \Theta)$
2. $e_{adv} = argmax(L(\bar x + \bar e; \hat \Theta))$ where norm of $\bar e$ less than ϵ

The parameters include:

1. $\hat \Theta$ is a constant copy of the current Θ;
2. $\bar e$ denotes the perturbations;

Explaination:

Task is to find $\bar e_{adv}$ that maximizes the classification loss. The $\bar e_{adv}$ can be regarded as the worst case of perturbations against the current classifier p(yn|Tn; Θ).

classification loss L_{cls}

$L_{cls} = L(E;\Theta)$

$L_{adv} = L(E+\bar e_{adv};\Theta)$

adversarial loss L_{adv}

$\hat L(\Theta) = \alpha L_{cls} + (1-\alpha) L_{adv}$

The parameter α in Equations (17) and (24) provides a handler for controlling the weights of classification loss and adversarial loss in loss functions.

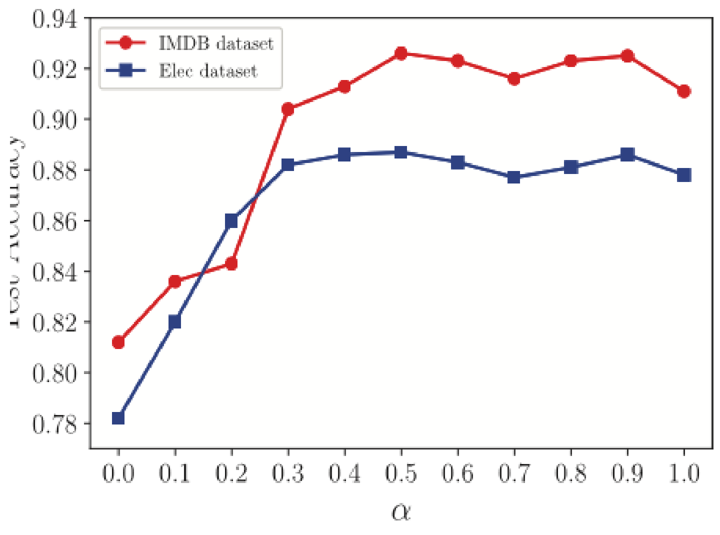


FIGURE 6 Test accuracy varies with α in traditional sentiment classification

xw

The parameter ϵ controls the threshold of selecting e_{adv}.

How to do with the adversarial learning.

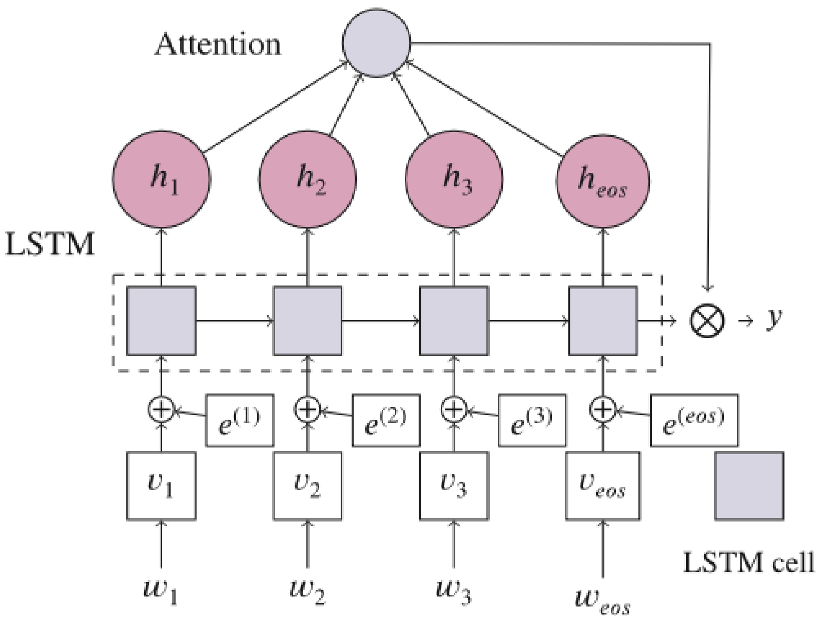


FIGURE 2 The built model for sentiment classification with perturbed word embeddings and attentive LSTM

where e(i) (i = 1, 2, ...) denotes the element in $\bar e_{adv} $. hi (i = 1, 2, ...) is the hidden output of each LSTM cell.

Experiments:

IMDB dataset, Elec dataset ,and SST-2 dataset are used to evaluate the performance of the proposed model.

Conclusion

give a comprehensive study of adversarial learning in sentiment classification.

The framework contains two models: sentiment clas- sification with adversarial learning (SCAT) and PU learning problem with adversarial learning (PUAT).

Adversarial learning brought our models more robustness by improving the quality of word embedding, especially for sentimental words.

Micro-blog sentiment classification using Doc2vec + SVM model with data purification

The main idea of the passage on "Micro-blog sentiment classification using Doc2vec + SVM model with data purification" is to present a methodology for improving sentiment classification on micro-blog content, such as tweets, by employing a combination of Doc2vec for feature extraction, Support Vector Machine (SVM) for classification, and data purification techniques to enhance the quality of the data used for training the model.

1. Doc2vec for Feature Extraction: Doc2vec is an extension of the Word2vec model that generates vector representations for documents, capturing the semantic meaning of words within their context

in the document. By using Doc2vec, the approach aims to convert micro-blog posts into meaningful, fixed-length feature vectors that represent the content and sentiment of the posts.

2. SVM for Classification: Support Vector Machine (SVM) is a powerful machine learning model used for classification tasks. After transforming the micro-blog posts into vectors using Doc2vec, the SVM model is trained on these vectors to classify the sentiment of the posts, typically into categories like positive, negative, or neutral.
3. Data Purification: Recognizing that the quality of training data significantly affects the performance of machine learning models, this approach includes a data purification step. Data purification involves preprocessing and cleaning the dataset to remove noise, irrelevant information, and any data that could introduce bias or inaccuracies into the model. This step is crucial for dealing with the informal language, slang, and abbreviations often found in micro-blog content, which can challenge sentiment analysis. The combination of these techniques aims to address the challenges of sentiment classification in micro-blog content, which includes dealing with short texts, informal language, and the dynamic nature of online language. By using Doc2vec to capture the semantic essence of posts, SVM for efficient and effective classification, and data purification to ensure the quality of the dataset, this methodology seeks to improve the accuracy and reliability of sentiment classification for micro-blog platforms.