Leveraging BERT for Enhanced Tweet Sentiment Analysis

Khanh Vu, Changling Li, Zihan Zhu, Xin Chen
Group: Attention is all you need
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—Large language models have revolutionized various fields, showcasing their capacity for understanding human natural languages. Tweet sentiment analysis is a challenging and valuable task within its context, given its direct relevance to social media analyses. In this report, we delve into a method utilizing BERT (Bidirectional Encoder Representations from Transformers) [1] ensemble, showcasing its efficiency in achieving commendable performance in tweet sentiment analysis compared to several standard baselines. We conduct a comprehensive suite of experiments, with discussions on the critical role of data preprocessing in improving model performance. Our findings provide insights into the development of more robust and efficient sentiment analysis models.**

I. Introduction

Tweet sentiment analysis has increasingly become an important task due to the surge in social media's role in shaping public opinion and societal trends. The development of accurate models for tweet sentiment analysis could potentially unlock insightful understandings. At the crux of our study is the application of large language models (LLMs), which have recently revolutionized the field of Natural Language Processing (NLP) with their impressive capability to comprehend and generate human-like text.

Historically, traditional machine learning methods like Support Vector Machines (SVM) [2], or deep learning architectures like Long Short-Term Memory (LSTM) [3] networks and Convolutional Neural Networks (CNN) [4], have been utilized for this task. However, they often fall short in capturing the intricate nuances of language, and lack the depth of understanding required for complex language constructs found in tweets. This is where LLMs like BERT (Bidirectional Encoder Representations from Transformers) [1] demonstrate their capability. With its transformer architecture, BERT provides deeper and broader comprehension of the context, allowing for a more nuanced understanding of sentiments in tweets.

In our approach, we employ an ensemble of BERT models, which collectively outperform every single model. We compare this ensemble model against baselines such as Multilayer Perceptron (MLP) [5], Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) [6]. We also implemented classical ML baselines like XGBoost [7] and Random Forest [8]. Our results underscore the ensemble model's efficacy, which improves accuracy by 9%-18% compared to baseline models. We also scrutinize how different data preprocessing strategies influence the outcomes. While we present a counter-intuitive result that the raw data works best with BERT, we provide discussions on why this might be the case and point out potential future research directions. Our contribution is three-fold:

- We provide an effective way of creating BERT model ensemble to improve tweet sentiment analysis.
- We showcase various data preprocessing methods and their effects, and provide discussions on their implications for LLM training.
- With the code submitted, we contribute to the open-source community of LLM and important baseline implementations for tweet sentiment analysis.

II. Background

Data pre-processing and feature extraction. Text sentiment classification necessitates two pivotal steps: data pre-processing, to sanitize noisy online text data, and feature extraction, to obtain suitable inputs for machine learning methods. Multiple studies have investigated various pre-processing techniques, such as stop-word removal [9], feature pruning [10], and a combination of techniques [11]. Common feature extraction techniques include the frequency of key words to biagram [12] or trigrams [13], presence of key word [14], and emoticon recognition [15].

Classification. Early sentiment classification works relied on statistical methods like Naive Bayes [16] and max-entropy based text classification [17], or traditional machine learning like SVM [14]. The advent of neural networks catalyzed the development of deep learning classifiers, combining CNN with pre-trained vectors for improved performance [18], or using RNN variants for document-level sentiment classification [19]. With the introduction of transformers, BERT and similar models have further elevated the classification performance [1].

III. Methods

A. Data preprocessing

Text preprocessing: We preprocess the original dataset using a combination of the following methods: (1) Cleaned: The raw dataset subjected to duplication removal, lowercasing, punctuation elimination, and Unicode error correction. (2) Lemmatization. (3) Removing stop words. (4) Spelling correction.

Loading the data: Our data loading procedure incorporates a special batching mechanism for higher GPU utilization. We cluster tweets of similar lengths, padding as necessary, to prevent the mixing of substantially disparate length tweets within the same batch. This approach eliminates inefficiencies that arise from processing shorter sequences in batches dominated by longer ones.

B. Use of Pretrained LLMs and Tokenizers

We fine-tune an ensemble of pretrained LLMs, provided by the Huggingface library, to perform sentiment analysis. Our ensemble includes bert-base-uncased [1], roberta-base, roberta-large [20], albert-base-v2, albert-large-v2, albert-xlarge-v2 [21], microsoft/deberta-large-v3 [22], cardiffnlp/twitter-roberta-base-sentiment-latest [23]. For properly tokenizing the data, we use (1) The BERT tokenizer, which employs WordPiece, to convert each word from the vocabulary into an embedding vector, and (2) the RoBERTa tokenizer, which uses byte-level Byte-Pair Encoding (BPE), akin to the GPT models, to provide an alternate method of tokenization.

To better fine-tune the LLMs for our task, we further attach an additional classification head after all pretrained layers. We implemented two different types of heads, catering different needs from the pretrained network. The first classifier head is a two-layer multilayer perceptron (MLP), with the input being the embedding of the special `[CLS]` token. To stabilize the training process and combat the potential instabilities associated with the vast representational space, layer normalization is applied prior to inputting the embedding into the MLP to further stabilize the gradients. Furthermore, to address the issue of overfitting, we incorporate multi-sample dropout [24]. This technique creates an ensemble-like effect during training and helps to smooth out the loss curve, providing a more stable and generalizable model.

The second classifier head takes a more intricate approach. Instead of relying on the `[CLS]` token's embedding alone, it operates on a combination of the last four hidden layers of the Transformers. These layers are concatenated, resulting in a hidden state of shape $(batch\_size, tweet\_length, h \times 4)$ with h denotes the dimension of token embeddings. Motivated by recent work in the field [25], we employ an attention mechanism to distill this rich, multidimensional representation into a single vector, capturing the most salient features in the sequence. This combined vector is then input into a single-layer MLP to produce the final logit. Through experimentation (Table II), we found that this attention-based head yields superior results over the MLP-based head and therefore stick to this classification head for all our fine-tuning experiments.

Model	# Params	Weight	Train accuracy	Val. accuracy
BERT	110M	-0.0713	0.9026	0.8923
RoBERTa (base)	125M	-0.0600	0.9049	0.8984
RoBERTa (large)	356M	0.4709	0.9207	0.9040
AlBERT (base)	12.7M	0.0368	0.9025	0.8964
AlBERT (large)	18.7M	0.1847	0.9119	0.8991
AlBERT (xlarge)	58.7M	0.3074	0.9223	0.9012
TimeLM	125M	0.2538	0.9035	0.9002
DeBERTa (large)	436M	0.8973	0.9207	0.9087
Ensemble (Ridge)				0.9133

Table I: Model size and performance results. The final model is an ensemble of eight models. The **Weight** column indicates the mixing weights used in the final ensemble.

Classification head	BERT	RoBERTa (base)	RoBERTa (large)
2-layer MLP	0.8849	0.8941	0.9012
Attention	0.8923 (↑ 0.74%)	0.8984 (↑ 0.43%)	0.9040 (↑ 0.28%)

Table II: Performance comparison between the two classification heads: 2-layer MLP vs. Attention. Validation accuracy's were reported.

C. Prediction thresholding and Ensemble learning

For individual model predictions, we exploited the validation set to search for the ideal decision threshold. We eventually arrived at an optimal threshold $t = 0.5$ for classifying a prediction as positive based on the predicted probability.

For the ensemble learning, we employed regularized logistic regression (Ridge) to seek for the best mixing weights which were then used to aggregate the predicted probabilities from different models into a single probability value. Here, we leveraged cross-validation to select the best hyperparameters and weights. Consequently, our final prediction emerged as an ensemble of eight fine-tuned models (Table I), offering a robust representation of collective model sentiment assessment.

IV. Experiments

A. Model Training Details

In our experiments, we carefully tuned our training configurations. AdamW is the selected optimizer, with a base learning rate set to 5e-5 and a weight decay of 0.3. The loss function utilized was binary cross-entropy (BCEWithLogitsLoss). Taking advantage of the common practice in transfer learning, we set different learning rates for each of the pretrained layers. Specifically, the last layer was assigned a learning rate of 5e-5, with each preceding layer's learning rate set to 90% of the subsequent layer's rate. This results in a geometrically decreasing learning rate across layers, adapting learning based on the layer hierarchy. We employed a learning rate scheduler which implements a linear warm-up phase for the first 6% of the total training steps, and thereafter follows a linear annealing schedule down to 0. The validation process was executed every quarter of the training epoch, allowing for frequent assessments of model performance throughout the training process. In total, each

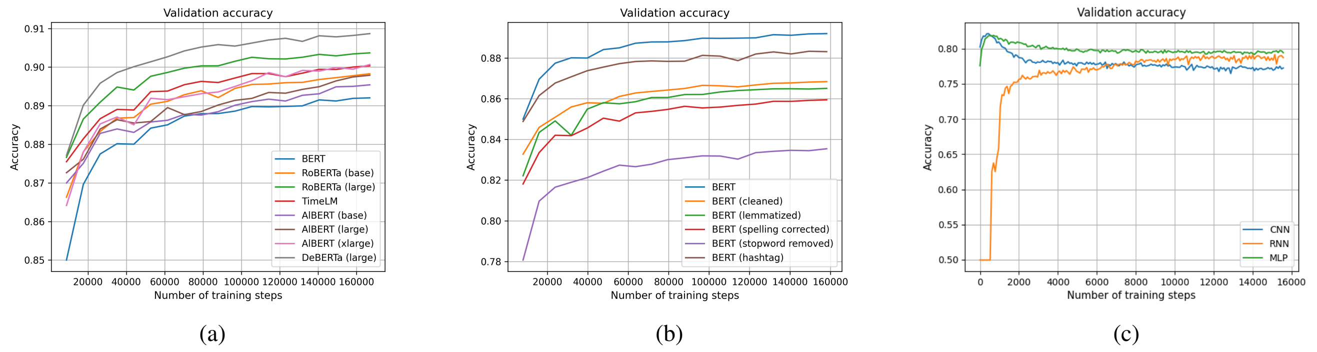

Figure 1: Validation performance for deep learning models. For BERT, we studied six data preprocessing strategies (b), and found that using the raw data performs the best.

model was being fine-tuned for 5 epochs on the 90% of the original train set (≈ 2.25M tweets) and being evaluated on the other 10%.

B. Baseline Implementation Details

We implemented 4 baselines from classical machine learning literature, using the scikit-learn [26] and XGBoost [7] library. For each method, we use three feature extraction and embedding techniques, namely count vectorization, TF-IDF vectorization [27], and GloVe embedding [28].

Logistic Regression [29]: This model is best suited for handling linearly separable data. We use the LogisticRegression method from scikit-learn, set the regularization parameter C to 1e5, and limit the maximum number of iterations to 100.

Random Forest [8]: We use the RandomForestClassifier from scikit-learn, which combines ensemble learning with decision trees. We set the number of estimators to 100, the maximum depth of each tree to 50, and the maximum number of features for each tree node to 50.

XGBoost [30]: XGBoost is an advanced gradient boosting algorithm that sequentially builds multiple weak learners. We set the number of estimators to 100 and the learning rate to 0.1, using the XGBClassifier from the XGBoost library.

Ridge Regression [31]: We use the default implementation of Ridge Regression in the scikit-learn package (sklearn.linear model.Ridge). We further add L2 regularization to avoid overfitting with the regularization strength set to 1.0. Since the default API cannot perform binary classification, we round the prediction output to 0 and 1.

We also compare our method with three deep learning models, MLP, CNN, and RNN. We adhere to the same data splitting process as our BERT ensemble method. For tokenization, we use keras.preprocessing.text library to determine word frequencies and identified the top 10,000 most frequently used words. We use zero-padding to extend each sequence to a length of 164 to standardize sequence lengths across tweets. For word embedding, we use the 100-dimensional GloVe Twitter embedding (glovetwitter27b). We introduce the most important training configurations for each model below, and more details can be found in our code in the supplementary materials.

MLP: We begin with a GloVe Embedding layer, flattened for 1D processing. Hidden layers of [64, 32, 16] neurons use ReLU activation for increased expressivity. Dropout layers (p=0.2) are used to avoid overfitting. The output layer uses sigmoid activation for binary classification. Our training uses binary cross-entropy loss and Adam optimizer with a learning rate of 1e-4.

CNN: Our CNN model initializes its first input layer with GloVe embedding weights. Following this are two layers of 1D convolution and max-pooling operations (pool size 5, stride 0), reducing dimensionality. The model is rounded off by a dense 2D layer (5, 16) and a flatten layer. The output layer employs a sigmoid activation for binary classification. The training uses binary cross-entropy loss and an Adam optimizer (learning rate 1e-4).

RNN: For RNN, we start with an input layer with fixed GloVe embedding weights. This is followed by a max-pooling layer (pool size 2), an LSTM layer (100 nodes, dropout rate 0.2, recurrent dropout rate 0.1), and a dense layer (16) with ReLU activation. The output layer uses sigmoid activation for binary classification. We train using binary cross-entropy loss and the Adam optimizer (learning rate 1e-4).

C. Results

We show each model's performance in Figure 1a, with more details in Table I and II. The results corroborate the existing observation for the ”scaling law” in large language models [32]; larger models consistently outperform their smaller counterparts. DeBERTa, the largest model, obtains the highest validation accuracy. AlBERT achieves decent results despite having smaller model size, suggesting its suitability for resource-constrained settings. When we use

	Logistic Regression			Random Forest			XGBoost			Ridge Regression		
	Original	Bigram	Remove Stopword	Original	Bigram	Remove Stopword	Original	Bigram	Remove Stopword	Original	Bigram	Remove Stopword
Count Vectorization	80.26	75.00	78.89	76.67	66.17	74.30	75.90	63.77	74.07	79.99	74.86	78.62
TF-IDF Vectorization	80.36	75.08	78.84	76.10	65.82	73.41	76.07	63.91	74.40	80.25	75.02	78.71
GloVe Embedding	76.39	76.39	74.54	76.39	75.26	73.29	75.20	78.41	76.67	78.41	76.17	74.40

Table III: Results of classical ML baselines.

Model	**# Params**	**Train accuracy**	**Val. accuracy**
MLP	12.6M	1.00	0.8188
CNN	11.6M	1.00	0.8213
RNN	11.6M	0.8633	0.7916

Table IV: DNN-based baseline classifier size and performance results.

this ensemble of models for our final predictions, we reach a validation accuracy of 0.91, surpassing every single model trained on the dataset.

Observations from Figure 1b revealed a somewhat counterintuitive phenomenon: The raw dataset was most effective for training BERT, surpassing conventional, meticulously cleaned datasets. The more we make changes to the original data (like removing stopwords), the worse performance we get in the end. The worst data preprocessing method, albeit sounding reasonable in theory, can lead to a 6% drop in the training performance, which can be more significant than creating a new large language model if we compare the performance variance of Figure 1a and Figure 1b. This unexpected outcome may be attributed to BERT's pretraining on a corpus that mirrors natural human language distribution. Hence, a dataset closely resembling this original distribution—like our raw dataset—might yield the best results. This finding prompts us to reconsider the perceived significance of comprehensive preprocessing and encourages further exploration into how data characteristics can influence tweet sentiment analysis in large language models. We provide more discussions in Section V.

For classical machine learning methods, we also observe that variants such as bigram or removing stopwords often yield inferior results compared to the original settings. TF-IDF vectorization coupled with logistic regression is among the strongest method, which yields a validation accuracy of 80.36%. Deep neural network (DNN) based methods offer a slight improvement in validation accuracies. It's noteworthy that both the MLP and CNN classifiers achieve commendable accuracy performances of 0.818 and 0.821 respectively, requiring fewer training iterations (Fig 1c). Conversely, the RNN classifier necessitates a significantly larger number of steps to reach a comparable performance of 0.791. For fair comparisons, we intentionally ran a lot more training steps for MLP and CNN for them to be compared with RNN, and found that prolonging the training leads to severe overfitting. This shows that we should early-stop MLP and CNN, while RNN requires a lot more computational resources to reach comparable results.

V. Discussion

One major takeaway from our experiments is that using the raw dataset performs better than applying many data preprocessing methods on large language models. We provide three hypotheses on why this might be the case, which can enlighten possible future research:

Contextual Understanding: BERT is designed to understand the context of each word in a sentence by looking at the words that come before and after it. This is particularly important in tweets, where the meaning of a word can heavily depend on its context. Preprocessing methods that remove or alter words (such as removing stopwords) could potentially disrupt this context, leading to a loss of information that BERT could otherwise use.

Handling of Informal Language: Tweets often contain informal language, slang, and abbreviations. Traditional preprocessing methods might remove or normalize these elements (such as spelling correction), but in the context of sentiment analysis, they can carry significant sentiment information. BERT's pretraining on a large corpus allows it to understand a wide variety of language styles and nuances, including the informal language often found in tweets.

Preservation of Original Syntax and Semantics: Some preprocessing methods can alter the original syntax and semantics of the text, such as stop words removal and lemmatization. Since BERT was pretrained on the book corpus and English Wikipedia where the langauges are natural languages, altering the syntax and semantics of a tweet might introduce unnecessary noise for it to understand a tweet's sentiment.

Although we have conducted extensive analysis on the task of understanding tweet sentiment, further studies should be carried out on a wider spectrum of tasks and models to generalize this conclusion - raw data best suits LLMs - for future engineering and research.

VI. Conclusion

In this study, we proposed an ensemble method based on the BERT model and its subsequent variants. Our method has demonstrated robust performance, outperforming traditional machine learning models by a substantial margin. Interestingly, we found that using raw data for BERT finetuning provided the best performance compared to standard preprocessing approaches. These findings serve as a vital step towards building more efficient, scalable, and effective sentiment analysis systems for social media data.

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

[2] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, “Support vector machines,” *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.

[3] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] K. O’Shea and R. Nash, “An introduction to convolutional neural networks,” *arXiv preprint arXiv:1511.08458*, 2015.

[5] L. Noriega, “Multilayer perceptron tutorial,” *School of Computing. Staffordshire University*, vol. 4, no. 5, p. 444, 2005.

[6] L. R. Medsker and L. Jain, “Recurrent neural networks,” *Design and Applications*, vol. 5, no. 64-67, p. 2, 2001.

[7] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, R. Mitchell, I. Cano, T. Zhou *et al.*, “Xgboost: extreme gradient boosting,” *R package version 0.4-2*, vol. 1, no. 4, pp. 1–4, 2015.

[8] V. Svetnik, A. Liaw, C. Tong, J. C. Culberson, R. P. Sheridan, and B. P. Feuston, “Random forest: a classification and regression tool for compound classification and qsar modeling,” *Journal of chemical information and computer sciences*, vol. 43, no. 6, pp. 1947–1958, 2003.

[9] H. Saif, M. Fernandez, Y. He, and H. Alani, “On stopwords, filtering and data sparsity for sentiment analysis of twitter,” 2014.

[10] M. Hu and B. Liu, “Mining opinion features in customer reviews,” in *AAAI*, vol. 4, no. 4, 2004, pp. 755–760.

[11] Z. Jianqiang and G. Xiaolin, “Comparison research on text pre-processing methods on twitter sentiment analysis,” *IEEE access*, vol. 5, pp. 2870–2879, 2017.

[12] B. Pang, L. Lee *et al.*, “Opinion mining and sentiment analysis,” *Foundations and Trends® in information retrieval*, vol. 2, no. 1–2, pp. 1–135, 2008.

[13] A. Pak, P. Paroubek *et al.*, “Twitter as a corpus for sentiment analysis and opinion mining.” in *LREc*, vol. 10, no. 2010, 2010, pp. 1320–1326.

[14] B. Pang, L. Lee, and S. Vaithyanathan, “Thumbs up? sentiment classification using machine learning techniques,” *arXiv preprint cs/0205070*, 2002.

[15] A. Go, R. Bhayani, and L. Huang, “Twitter sentiment classification using distant supervision,” *CS224N project report, Stanford*, vol. 1, no. 12, p. 2009, 2009.

[16] C. Manning and H. Schutze, *Foundations of statistical natural language processing*. MIT press, 1999.

[17] K. Nigam, J. Lafferty, and A. McCallum, “Using maximum entropy for text classification,” in *IJCAI-99 workshop on machine learning for information filtering*, vol. 1, no. 1. Stockholom, Sweden, 1999, pp. 61–67.

[18] Y. Kim, “Convolutional neural networks for sentence classification,” *arXiv preprint arXiv:1408.5882*, 2014.

[19] D. Tang, B. Qin, and T. Liu, “Document modeling with gated recurrent neural network for sentiment classification,” in *Proceedings of the 2015 conference on empirical methods in natural language processing*, 2015, pp. 1422–1432.

[20] 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.

[21] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “Albert: A lite bert for self-supervised learning of language representations,” 2020.

[22] P. He, X. Liu, J. Gao, and W. Chen, “Deberta: Decoding-enhanced bert with disentangled attention,” 2021.

[23] D. Loureiro, F. Barbieri, L. Neves, L. E. Anke, and J. Camacho-Collados, “Timelms: Diachronic language models from twitter,” 2022.

[24] H. Inoue, “Multi-sample dropout for accelerated training and better generalization,” 2020.

[25] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, “A structured self-attentive sentence embedding,” 2017.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, “Scikit-learn: Machine learning in python,” *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[27] K. Sparck Jones, “A statistical interpretation of term specificity and its application in retrieval,” *Journal of documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[28] J. Pennington, R. Socher, and 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.

[29] R. E. Wright, “Logistic regression.” 1995.

[30] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[31] A. E. Hoerl and R. W. Kennard, “Ridge regression: Biased estimation for nonorthogonal problems,” *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[32] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, “Scaling laws for neural language models,” *arXiv preprint arXiv:2001.08361*, 2020.

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):

Leveraging BERT for Enhanced Tweet Sentiment Analysis

Authored by (in block letters):

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

Name(s):	**First name(s):**
Vu	Khanh
Li	Changling
Zhu	Zihan
Chen	Xin

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	Khanh
	Changling Li
	[illegible]
	Xin Chen

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