Ulm University | 89069 Ulm | Germany

Faculty for Engineering, Computer Science and Psychology
Institute of Databases and Information Systems (DBIS)

Transformers are Short Text Classifiers: A Study of Inductive Short Text Classifiers on Benchmarks and Real-world Datasets

Bachelor Thesis at Ulm University

Presented by:
Fabian Karl
fabian.karl@uni-ulm.de
1046415

Supervisors:
Prof. Dr.-Ing. Ansgar Scherp

2022

Version November 28, 2022

Contents

List of Figures

List of Tables

Transformers are Short Text Classifiers: A Study of Inductive Short Text Classifiers on Benchmarks and Real-world Datasets

Fabian Karl
fabian.karl@uni-ulm.de
University of Ulm
Germany

ABSTRACT

Short text classification is a crucial and challenging aspect of Natural Language Processing. For this reason, there are numerous highly specialized short text classifiers. However, in recent short text research, State of the Art (SOTA) methods for traditional text classification, particularly the pure use of Transformers, have been unexploited. In this work, we examine the performance of a variety of short text classifiers as well as the top performing traditional text classifier. We further investigate the effects on two new real-world short text datasets in an effort to address the issue of becoming overly dependent on benchmark datasets with a limited number of characteristics. Our experiments unambiguously demonstrate that Transformers achieve SOTA accuracy on short text classification tasks, raising the question of whether specialized short text techniques are necessary.

CCS CONCEPTS

• **Computing methodologies → Classification and regression trees**; **Supervised learning by classification**; **Natural language processing**; • **Information systems → Clustering and classification**; *Language models.*

KEYWORDS

Text Classification, Natural Language Processing, Short Text, Transformer, BERT, GNN

1 INTRODUCTION

Text classification is a crucial aspect of Natural Language Processing (NLP), and extensive research in this field is being conducted. Many researchers are working to improve the speed, accuracy, or robustness of their algorithms. Traditional text classification, however, does not take some traits into account that appear in numerous real-world applications, such as short text. Therefore, studies have been conducted specifically on short texts [34, 42]. From user-generated content like social media to business data like accounting records, short text covers a wide range of topics. For example, the division into goods and services is an important part of the tax audit. Currently, an auditor checks whether the element descriptions match the appropriate class of good or service. Since this can be very time-consuming, it is desirable to facilitate this work and to bring it into a semi-automatic context with the help of classifiers. Also, the subdivision into more specific classes can be useful for determining whether a given amount for an entry is reasonable.

Since short texts are typically only one to two sentences long, they lack context and therefore pose a challenge for text classification. In order to get better results, many short text classifiers also operate in a transductive setup [34, 37, 39], which includes the test set during training. However, as they need to be retrained each time new data needs to be classified, those transductive models are not very suitable for real-world applications. The results of both transductive and the generally more useful inductive short text classifier are typically unsatisfactory due to the challenge that short text presents. Recent studies on short texts have emphasized specialized models [29, 32, 34, 37, 39, 42] to address the issues associated with the short text length. However, State of the Art (SOTA) text classification methods, particularly the pure use of Transformers, have been unexploited. In this work, the effectiveness on short texts is examined and tested by means of benchmark datasets. We also introduce two new, realistic datasets in the domain of goods and services descriptions.

Our work includes the following contributions:

- We provide a comparison of various modern text classification techniques. In particular, specialized short text methods are compared with the top performing traditional text classification models.
- To cover additional characteristics and test the models in a realistic use-case, we also introduce two new real-world datasets in the goods and services domain.
- We demonstrate that Transformers achieve SOTA accuracy on short text classification tasks. questioning the need of specialized short text classifier.

Below, we summarize the related work. Section 3 provides a detailed description of the models that were selected for our experiments. The experimental apparatus is described in Section 4. An overview of the achieved results is reported in Section 5. Section 6 discusses the results, before we conclude.

2 RELATED WORK

Despite the fact that Bag of Words (BoW)-based models have long represented the cutting edge in text classification, attention has recently shifted to sequence-based and, more recently, graph-based concepts. However, BoW-based models continue to offer a solid baseline [6]. For example in fastText [11] the average of the trained word representations are used as text representation and then fed into a linear classifier. This results in an efficient model for text classification. To give an overview of the various concepts, Section 2.1 provides various works in the field of sequence-based models, Section 2.2 discusses graph-based models, and Section 2.3 examines how these concepts are applied to short text. Finally, a summary of the findings from the related work is presented in Section 2.4.

2.1 Sequence-based Models

For any NLP task, Recurrent Neural Networks (RNN) and Long short-term memory (LSTM) are frequently used and a logical choice because both models learn historical information while taking location information for all words into account [14, 20]. Since RNNs must be computed sequentially and cannot be computed in parallel, the use of Convolutional Neural Networks (CNNs) is also common [14, 30]. The text must be represented as a set of vectors that are concatenated into a matrix in order to be used by CNNs. The standard CNN convolution and pooling operations can then be applied to this matrix. TextCNN [12] uses this in combination with pretrained word embeddings for sentence-level classification tasks. While CNN-based models extract the characteristics from the convolution kernels, the relationship between the input words is captured by RNN-based models [14]. An important turning point in the advancement of NLP technologies was the introduction of Bidirectional Encoder Representations from Transformers (BERT) [31]. By performing extensive pre-training in an unsupervised manner and automatically mining semantic knowledge, BERT learns to produce contextualized word vectors that have a global semantic representation. BERT-like models are suited for large datasets as they can parallelize computation [14]. The effectiveness of BERT-like models in text classification is demonstrated by Galke and Scherp [6]. However, very few works take them into account when classifying texts.

2.2 Graph-based Models

Recently, text classification has paid a lot of attention to graph-based models, particularly Graph Neural Networks (GNNs) [3, 25, 33]. This is due to the fact that tasks with rich relational structures benefit from the powerful representation capabilities of GNNs, which preserve global structure information [33]. The task of text classification offers this rich relational structure because text can be modeled as edges and nodes in a graph structure. There are different ways to represent the documents in a graph structure, but two main types have emerged [33, 34]. The first approach builds a graph for each document using words as nodes and structural data, such as word co-occurence data, as edges. However, only local structural data is used. The task is constructed as a whole graph classification problem in order to classify the text. A popular *document-level* approach is HyperGAT [5] which uses a dual attention mechanism and hypergraphs applied to documents to learn text embeddings. The second approach creates a graph for the entire corpus using words and documents as nodes. The text classification task is now a node classification task for the unlabeled document nodes. The drawback of this method is that models using it are inherently transductive. For example, TextGCN [38] uses this concept by employing a standard Graph Convolutional Networks (GCN) on this heterogeneous graph. Following TextGCN, Lin et al. [16] propose BertGCN, a model that makes use of BERT to initialize representations for the document nodes in order to combine the benefits of both the large-scale pretraining of BERT and the transductive TextGCN. However, the increase provided by this method is limited to datasets with long average text lengths. Zeng et al. [40] also experiment with combining TextGCN and BERT in the form of TextGCN-Bert-serial-SB, a Simplified-Boosting Ensemble, where BERT is only trained on the TextGCN's misclassification. Which model is applied to which document is determined by a heuristic based on the node degree of the test document. However, TextGCN-CNN-serial-SB, which substitutes TextCNN for BERT, yields better results. Another approach of combining graph classifiers with BERT-style models is ContTextING [10]. By using a joint training mechanism, TextING [41] and BERT are trained on sub-word tokens and base their predictions on the results of the two models. As opposed to applying each model separately, this produces better results.

2.3 Short Text Models

Of course, short texts can also be classified using the methods discussed above. However, this is challenging because short texts tend to lack context and adhere to less strict syntactic structure [34]. This has led to the emergence of specialized techniques that focus on improving the results for short text. For instance, Heterogeneous Graph Attention networks (HGAT) [37] is a powerful semi-supervised short text classifier. This was the first attempt to model short texts as well as additional information like topics gathered from a Latent Dirichlet Allocation (LDA) [1] and entities retrieved from Wikipedia with a Heterogeneous Information Network (HIN). To achieve this, a HIN embedding with a dual-level attention mechanism for nodes and their relations was used. For the semantic sparsity of short text, both the additional information and the captured relations are beneficial. A transductive and an inductive HGAT model were released, with the transductive model being better on every dataset. By adapting neighbor contrastive learning from Hu et al. [9], NC-HGAT [29] expands the HGAT model to produce a more robust variant. Neighbor contrastive learning is based on the premise that documents that are connected have a higher likelihood of sharing a class label and, as a result, should therefore be closer in feature space. In order to represent the additional information, SHINE [34] also makes use of a heterogenous graph. In contrast, SHINE generates component graphs in the form of word, entity, and Part Of Speech (POS) graphs and creates a dynamically learned short document graph by employing hierarchical pooling over all component graphs. In the semi-supervised setting, SHINE outperforms HGAT as a strong transductive model. Short-Text Graph Convolutional Networks (STGCN) [39] is an additional short text classifier. A graph of topics, documents, and unique words is the foundation of STGCN. Although the STGCN results by themselves are not particularly impressive, the impact of pre-trained word vectors obtained by BERT was also examined. The classification of the STGCN is significantly enhanced by the combination of STGCN, BERT, and a Bi-LSTM.

2.4 Summary

Graph neural network-based methods are widely used in short text classification. However, in recent short text research, SOTA text classification methods, particularly the pure use of Transformers, have been unexploited. Even though STGCN uses BERT to enhance their outcomes, the results of their fine-tuned BERT demonstrate that they did not fully utilize BERT's potential. Additionally, the majority of short text models are transductive. The crucial drawback of being transductive is that every time new data needs to be classified, the model must be retrained.

3 MODELS FOR TEXT CLASSIFICATION

This section provides a detailed description of the conceptual designs of the inductive models that were selected for our experiments. We focus on inductive models since it resembles a more realistic real-world scenario. We begin with models for short text classification in Section 3.1 and then Section 3.2 introduces a selection of top-performing models for text classification.

3.1 Models for Short Text Classification

The models listed below either make claims about their ability to categorize short texts or were designed with that specific goal.

3.1.1 SECNN. SECNN [32] is a text classification model built on CNNs that was created specifically for short texts with few and insufficient semantic features. Wang et al. [32] suggested four parts to address this issue. In order to achieve better coverage on the word vector table, they used an improved Jaro-Winkler similarity during preprocessing to identify any potential spelling mistakes. Second, they use a CNN model built on the attention mechanism to look for words that are related. Third, in order to accomplish the goal of short text semantic expansion, the external knowledgebase *Probase* [35] is used to enhance the semantic features of short text. Finally, the classification process is performed using a straightforward CNN with a Softmax output layer.

3.1.2 SGNN/ESGNN/C-BERT. Sequential Graph Neural Network (SGNN) [42] is a GNN-based model that emphasizes the propagation of features based on sequences. By training each document as a separate graph, it is possible to learn the words' local and sequential features. GloVe's [21] pre-trained word embedding is utilized as a semantic feature of words. In order to update the feature matrix for each document graph, a Bi-LSTM is used to extract the contextual feature of each word. After that, a simplified GCN aggregates the features of neighboring word nodes. Additionally, Zhao et al. [42] introduce two variants: Extended-SGNN (ESGNN), in which the initial contextual feature of words is preserved and C-BERT, in which the Bi-LSTM is swapped for BERT.

3.1.3 DADGNN. Deep Attention Diffusion Graph Neural Network (DADGNN) [18] is a graph-based method that combats the oversmoothing problem of GNNs and allows stacking more layers by utilizing attention diffusion and decoupling techniques. This decoupling technique is also very advantageous for short texts because it obtains distinct hidden features in deep graph networks.

3.1.4 Bi-LSTM. The Long short-term memory (LSTM) [8], which is frequently used in text classification, has a bidirectional variant called Bi-LSTM [17]. Due to its strong results for short texts [20, 42] and years of use as the SOTA method for many tasks, this model is a good baseline for our purpose.

3.2 Top-performing Models for Text Classification

An overview of the top text classification models that excel on texts of all lengths and were not specifically created with short texts in mind is provided in this section. We employ the base models for the Transformers.

3.2.1 BERT. Bidirectional Encoder Representations from Transformers (BERT) [4] is a language representation model that is based on the Transformer architecture [31]. Encoder-only models, such as BERT, rely solely on the encoder component of the Transformer architecture, whereby the text sequences are converted into rich numerical representations [30]. These models are well suited for text classification due to this representation. BERT is designed to incorporate a token's left and right contexts into its computed representation. This is commonly referred to as bidirectional attention.

For pre-training, BERT uses two unsupervised objectives:

(1) Masked Language Modeling (MLM): In this method, a percentage of the input tokens are masked at random with a [MASK] token, and the training objective is to predict those masked tokens. In order to counteract the effect of causing a mismatch between pre-training and fine-tuning, "masked tokens" are not always replaced by [MASK].
"Masked tokens" are:
    - 80% of the time replaced by [MASK]
    - 10% of the time replaced by a random token
    - 10% of the time left unmodified
(2) Next Sentence Prediction (NSP): The process of determining whether a given sentence is the next sentence after another is known as next sentence prediction. The input for this objective is a pair of two sentences separated by a [SEP] token, with 50% of the training data consisting of random second sentences and not the subsequent one. This aims to improve understanding of the relationship between two sentences, which language modeling does not directly capture.

3.2.2 RoBERTa. The Robustly optimized BERT approach (RoBERTa) [19] is a systematically improved BERT adaptation. In the RoBERTa model, the pre-training strategy was changed and training was done on larger batches with more data, to increase BERT's performance.

The following are the differences from BERT:

(1) Dynamic masking: While masking was done once during preprocessing in BERT, RoBERTa masks sequences dynamically as they are fed into the model. This is significant because static masking requires duplicated training data to obtain different maskings, which is impractical for large data sets.
(2) FULL-SENTENCES without NSP loss: Instead of using the NSP task, RoBERTa fills the input with full sentences that are sampled coherently until the total length reaches 512 tokens to learn long-range dependencies.
(3) Large mini-batches: Liu et al. [19] notice that training with large batches enhances perplexity and end-task accuracy for the masked language modeling target. Therefore, they raise the batch size to 8K.
(4) Larger byte-level Byte-Pair Encoding (BPE): BERT's 30K BPE vocabulary is replaced by a larger byte-level BPE vocabulary with 50K subword units, following the idea of Radford et al. [23]. This adds approximately 15M parameters and decrease end-task performance slightly, but it provides the benefit of a universal encoding scheme.

3.2.3 DeBERTa. To improve BERT and RoBERTa models, Decoding-enhanced BERT with disentangled attention (DeBERTa) [7] makes two architectural adjustments. The first is the disentangled attention mechanism, which encodes the content and location of each word using two vectors. The content of the token at position i is represented by H_i and the relative position $i|j$ between the token at position i and j are represented by $P_{i|j}$. The equation for determining the cross attention score is as follows:

$$A_{i,j} = H_i H_j^T + H_i P_{j|i}^T + P_{i|j} H_j^T + P_{i|j} P_{j|i}^T \tag{1}$$

The second adjustment is an enhanced mask decoder that uses absolute positions in the decoding layer to predict masked tokens during pre-training. For masked token prediction, DeBERTa includes the absolute position after the transform layers but before the softmax layer. In contrast, BERT incorporates the position embedding into the input layer. As a result, DeBERTa is able to capture the relative position in all Transformer layers.

3.2.4 ERNIE 2.0. Sun et al. [28] proposed ERNIE 2.0, a continuous pre-training framework that builds and learns pre-training tasks through continuous multi-task learning. This allows the extraction of additional valuable lexical, syntactic, and semantic information in addition to co-occurring information, which is typically the focus.

3.2.5 DistilBERT. The concept behind DistilBERT [24] is to leverage knowledge distillation to produce a more compact and faster version of BERT while retaining most of its language understanding capacities. DistilBERT reduces the size of BERT by 40%, is 60% faster, and still retains 97% of its language understanding capabilities. In order to accomplish this, DistilBERT optimizes the following three objectives while using the BERT model as a teacher:

(1) Distillation loss: The model was trained to output probabilities equivalent to those of the BERT base model. The loss function is defined as cross-entropy over the soft target probabilities

$$L_{ce} = \sum_i t_i \cdot \log(s_i) \tag{2}$$

where t_i and s_i are the probabilities estimated by the teacher and the student, respectively.

(2) Masked Language Modeling (MLM): As described by Devlin et al. [4] for the BERT model (See Section 3.2.1).

(3) Cosine embedding loss: Additionally, the model was trained to align the DistilBERT and BERT hidden state vector, as close as possible.

3.2.6 ALBERTv2. A Lite BERT (ALBERT) [13] is a Transformer that uses two parameter-reduction strategies to save memory and speed up training by sharing the weights of all layers across its Transformer. This model is therefore particularly effective for longer texts. During pretraining, ALBERTv2 employs MLM and Sentence-Order Prediction (SOP), which predicts the sequence of two subsequent text segments.

3.2.7 WideMLP. WideMLP [6] is a BoW-Based Multilayer Perceptron (MLP) with a single wide hidden layer of 1, 024 Rectified Linear Units (ReLUs). This model serves as a useful benchmark against which we can measure actual scientific progress.

3.2.8 InducT-GCN. InducTive Graph Convolutional Networks for Text classification (InducT-GCN) [33] is a GCN-based method that categorically rejects any information or statistics from the test set. To achieve the inductive setup, InducT-GCN represents document vectors with a weighted sum of word vectors and apply TF-IDF weights instead of representing document nodes with one-hot vectors. A two-layer GCN is employed for training, with the first layer learning the word embeddings and the second layer in the dimension of the dataset's classes outputs into a softmax activation function.

4 EXPERIMENTAL APPARATUS

First, the datasets used in our study are presented. Then we describe the preprocessing steps and our procedure. The selection and optimization of our hyperparameters are covered next. Finally, the experimentation's measures will be described.

4.1 Datasets

We employed a variety of datasets, which are further explained in this section, to demonstrate the performance of the various short text classification models. The key characteristics are denoted in Table 1. First, we describe the benchmark datasets. Second, we introduce our new datasets in the domain of goods and services.

4.1.1 Benchmark Datasets. Six short text benchmark datasets, namely R8, MR, SearchSnippets, Twitter, TREC, and SST-2, are used in our experiments. The following gives a detailed description of them.

- **R8** is an 8-class subset of the Reuters 21578 news dataset[1]. It is not a classical short text scenario with an average length of 65.72 tokens but offers the ability to set the methods in comparison to traditional text classification.
- **MR**[2] is a widely used dataset for text classification. It contains movie-review documents with an average length of 20.39 tokens and is therefore suitable for short text classification.
- The dataset **SearchSnippets**[3], which is made up of snippets returned by a search engine and has an average length of 18.10 tokens, was released by Phan et al. [22].
- **Twitter**[4] is a collection of 10, 000 tweets that are split into the categories negative and positive based on sentiment. The length of those tweets is on average 11.64 tokens.
- **TREC**[5], which was introduced by Li and Roth [15], is a question type classification dataset with six classifications for questions. It provides the shortest texts in our collection of benchmark datasets, with an average text length of 10.06 tokens.
- **SST-2**[6] [26] or SST-binary is a subset of the Stanford Sentiment Treebank, a fine-grained sentiment analysis dataset, in which neutral reviews have been removed and the data has either a positive or negative label. The average number of tokens in the texts is 20.32.

4.1.2 Goods and Services Datasets. In order to evaluate the performance on data with real world applications, we introduce two new datasets that are focused on the distinction between goods and services. Although there are already datasets for product classification, such as the WDC-LSPM[7], to the best of our knowledge, our datasets are the first to combine goods and services.

NICE is a classification system for goods and services that divides them into 45 classes and is based on the Nice Classification[8] of the World Intellectual Property Organization (WIPO). There are 11 classes for various service types and 34 categories for goods. With 9, 593 documents, NICE is comparable in size to the benchmark datasets. This dataset, which has texts with an average length of 3.75 tokens, is an excellent example of extremely short text. For the division into goods and services, there is also the binary version NICE-2.

Short Texts Of Products and Services (STOPS) is the second dataset we offer. With 200, 341 documents and an average length of 5.64 tokens, STOPS is a reasonably large dataset. The data set was directly derived from a potential use case in the form of Amazon descriptions and Yelp business entries, making it the most realistic. Like NICE, STOPS has a binary version STOPS-2.

Both datasets provide novel characteristic properties that the benchmark datasets did not cover. In particular, the abundance of fine-granular classes presents a new challenge that is not addressed by common benchmarks.

4.2 Preprocessing

The preprocessing used to produce NICE and STOPS will be covered in this section.

[1]http://www.daviddlewis.com/resources/testcollections/reuters21578/
[2]https://www.cs.cornell.edu/people/pabo/movie-review-data/
[3]http://jwebpro.sourceforge.net/data-web-snippets.tar.gz
[4]https://www.nltk.org/howto/twitter.html#Using-a-Tweet-Corpus
[5]https://cogcomp.seas.upenn.edu/Data/QA/QC/
[6]https://nlp.stanford.edu/sentiment/
[7]http://webdatacommons.org/largescaleproductcorpus/
[8]https://www.wipo.int/classifications/nice/en/

Table 1: Characteristics of short text classification datasets

Benchmark datasets	#Doc	#Train	#Test	#Classes	Avg. length
R8	7,674	5,485	2,189	8	65.72
MR	10,662	7,108	3,554	2	20.39
SearchSnippets	12,340	10,060	2,280	8	18.10
Twitter	10,000	7,000	3,000	2	11.64
TREC	5,952	5,452	500	6	10.06
SST-2	9,613	7,792	1,821	2	20.32
Goods and Services datasets	**#Doc**	**#Train**	**#Test**	**#Classes**	**Avg. length**
NICE-45	9,593	6,715	2,878	45	3.75
NICE-2	9,593	6,715	2,878	2	3.75
STOPS-41	200,341	140,238	60,103	41	5.64
STOPS-2	200,341	140,238	60,103	2	5.64

To create NICE, the WIPO[9] classification data was converted to lower case, all punctuation was removed, and side information that was enclosed in brackets was also removed. Additionally, accents were dropped. Following a random shuffle, the data was divided into 70% train and 30% test.

As product and service entries for STOPS, the product description of MAVE[10] [36] and the business names of YELP[11] were used. Due to the different data sources, these also had to be preprocessed differently. All classes' occurrences in the MAVE data were counted, and $5,000$ sentences from each of the 20 most common classes were chosen. The multi-label categories for the YELP data were broken down into a list of single label categories, and the sentences were then mapped to the most common single label that each one has. In order to prevent any label from taking up too much of the dataset, the data was collected such that there is a maximum of $1,200$ documents per label. After that, all punctuation was dropped, the data was converted to lower case, and accents were also dropped. The data was split into train and test in a $70:30$ ratio after being randomly shuffled.

There are scripts on GitHub[12] for recreating these datasets.

4.3 Procedure

The best short text classifier and text classification models were retrieved from the literature, and the accuracy scores were extracted in order to establish a comparison. Own experiments, particularly those using various Transformers, were conducted in order to compare them. Investigations into the impacts of hyperparameters on short texts were performed. More details about these are provided in Section 4.4. In order to test the methods in novel contexts, we also created two new datasets, whereby STOPS stands out due to its much higher quantity of documents.

4.4 Hyperparameter Optimization

Our experiments for BERT, DistilBERT, and WideMLP used the hyperparameter from Galke and Scherp [6]. The parameters for BERT and DistilBERT are a learning rate of $5 \cdot 10^{-5}$, a batch size of 128, and fine-tuning for 10 epochs. WideMLP was trained for 100 epochs with a learning rate of 10^{-3}, a batch size of 16, and a dropout of 0.5. For ERNIE 2.0 and ALBERTv2, we make use of the SST-2 values that Sun et al. [28] and Lan et al. [13], respectively, published. For our hyperparameter selection for DeBERTa and RoBERTa, we used the BERT values from Galke and Scherp [6] as a starting point and investigated the effect of smaller learning rates. The exact procedure is described in Appendix A. This resulted in learning rates of $2 \cdot 10^{-5}$ for DeBERTa and $4 \cdot 10^{-5}$ for RoBERTa while maintaining the other parameters. For comparison, we followed the same procedure to create ERNIE 2.0 (optimized), which yields a learning rate of $25 \cdot 10^{-6}$. The Bi-LSTM values from Zhao et al. [42] were used for both the LSTM and the Bi-LSTM model. We used DADGNN with the default parameters of 0.5 dropout, 10^{-6} weight decay, and two attention heads for all datasets.

4.5 Metrics

Accuracy is the metric used in the experiments to measure the classification of short text. The formula is as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

where TP is the number of positive classes and classified to be positive classes, TN is the number of negative classes and classified to be negative classes. FP and FN are the numbers of wrongly classified positive or negative classes, respectively. For multi-class cases, the subset accuracy is calculated.

5 RESULTS

The accuracy scores for the text classification models on the six benchmark datasets are shown in Table 2. The findings demonstrate that the relatively straightforward models LSTM, Bi-LSTM, and WideMLP provide a strong baseline across all datasets. This comparison clearly demonstrates the limitations of some models, with InducT-GCN falling short in all datasets except SearchSnippets, SECNN underperforming on TREC, and DADGNN producing weak MR results in our own experiment. The Transformer models, on the other hand, are the best performing across all datasets with the exception of SearchSnippets. With consistently strong performance across all datasets, DeBERTa stands out in particular. The graph-based models from Zhao et al. [42], SGNN, ESGNN, and CBERT, all perform well for the datasets for which results are available and ESGNN even outperforms all other models for SearchSnippets. Zhao et al. [42] used a modified training split and additional preprocessing. While an increase of about 5 percentage points for MR could be obtained by extending ESGNN with BERT in C-BERT, the increase is not noticeable for other datasets. When applied to short texts, the inductive models even outperform transductive models. On Twitter, they reach a performance of 99.97%, and when using BERT on the TREC

[9]https://www.wipo.int/nice/its4nice/ITSupport_and_download_area/20220101/MasterFiles/index.html
[10]https://github.com/google-research-datasets/MAVE
[11]https://www.yelp.com/dataset/download
[12]https://github.com/FKarl/Bachelor-Thesis

Table 2: **Accuracy on short text classification datasets. The "Short Text" column indicates whether the model makes claims about its ability to categorize short texts. Provenance refers to the source of the accuracy scores.**

Inductive Models	**Short Text**	**R8**	**MR**	**SearchSnippets**	**Twitter**	**TREC**	**SST-2**	**Provenance**
Transformer Models								
BERT	N	98.17[1]	86.94	88.20	99.96	**99.4**	91.37	Own experiment
RoBERTa	N	98.17[1]	89.42	85.22	99.9	98.6	94.01	Own experiment
DeBERTa	N	98.45[1]	**90.21**	86.14	99.93	98.8	**94.78**	Own experiment
ERNIE 2.0	N	98.04[1]	88.97	89.12	**99.97**	98.8	93.36	Own experiment
ERNIE 2.0 (optimized)	N	98.17[1]	89.53	89.17	**99.97**	99	94.07	Own experiment
DistilBERT	N	97.98[1]	85.31	89.69	99.96	99	90.49	Own experiment
ALBERTv2	N	97.62	86.02	87.68	**99.97**	98.6	91.54	Own experiment
BoW Models								
WideMLP	N	96.98	76.48	67.28	99.86	97	82.26	Own experiment
fastText	N	96.13	75.14	88.56[4]	–	–	–	Zhao et al. [42]
Graph-based Models								
HGAT[2]	Y	–	62.75	82.36	63.21	–	–	Yang et al. [37]
NC-HGAT[2]	Y	–	62.46	–	63.76	–	–	Sun et al. [29]
SGNN[3]	Y	98.09	80.58	90.68[4]	–	–	–	Zhao et al. [42]
ESGNN[3]	Y	98.23	80.93	**90.80**[4]	–	–	–	Zhao et al. [42]
C-BERT (ESGNN + BERT)[3]	Y	98.28	86.06	90.43[4]	–	–	–	Zhao et al. [42]
DADGNN	Y	98.15	78.64	–	–	97.99	84.32	Liu et al. [18]
DADGNN	Y	97.28	74.54	84.91	98.16	97.54	82.81	Own experiment
HyperGAT	N	97.97	78.32	–	–	–	–	Ding et al. [5]
InducT-GCN	N	96.67	75.25	76.68	88.53	92.42	79.98	Own experiment
ConTextING-BERT	N	97.91	86.01	–	–	–	–	Huang et al. [10]
ConTextING-RoBERTa	N	98.13	89.43	–	–	–	–	Huang et al. [10]
CNN and LSTMs								
SECNN[3]	Y	–	83.89	–	–	91.34	87.37	Wang et al. [32]
LSTM (BERT)	Y	94.28	75.10	65.13	99.83	97	81.38	Own experiment
Bi-LSTM (BERT)	Y	95.52	75.30	66.79	99.76	97.2	80.83	Own experiment
LSTM (GloVe)	Y	96.34	74.99	67.67	95.23	97.4	79.95	Own experiment
Bi-LSTM (GloVe)	Y	96.84	75.32	68.15	95.53	97.2	80.17	Own experiment
Bi-LSTM (GloVe)	Y	96.31	77.68	84.81[4]	–	–	–	Zhao et al. [42]
Transductive Models	**Short Text**	**R8**	**MR**	**SearchSnippets**	**Twitter**	**TREC**	**SST-2**	**Provenance**
Graph-based Models								
SHINE[5]	Y	–	64.58	82.39	72.54	–	–	Wang et al. [34]
SHINE	Y	86.48	63.21	83.29	71.49	79.90	62.56	Own experiment
STGCN	Y	97.2	78.2	–	–	–	–	Ye et al. [39]
STGCN+BiLSTM	Y	–	78.5	–	–	–	–	Ye et al. [39]
STGCN+BERT+BiLSTM	Y	98.5	82.5	–	–	–	–	Ye et al. [39]
TextGCN	N	97.07	76.74	83.49	–	–	–	Zhao et al. [42]
TextGCN	N	97.07	76.74	–	–	91.40	81.02	Liu et al. [18]
BertGCN	N	98.1	86.0	–	–	–	–	Lin et al. [16]
RoBERTaGCN	N	98.2	89.7	–	–	–	–	Lin et al. [16]
TextGCN-BERT-serial-SB	N	97.78	86.69	–	–	–	–	Zeng et al. [40]
TextGCN-CNN-serial-SB	N	**98.53**	87.59	–	–	–	–	Zeng et al. [40]

[1] With a batch size of 32 and for DeBERTa of 16.
[2] With only 40 randomly selected documents per class.
[3] Not reproducible. Authors have been contacted twice without a response.
[4] Using a modified training split of 8, 636 training and 3, 704 test documents and further preprocessing.
[5] Employing very low train ratios (0.38% to 6.22%).

dataset, a performance of 99.4%. Non-Transformer models also perform well on TREC, although Transformers outperform them.

The accuracy results for our newly introduced datasets, NICE and STOPS, are shown in Table 3. New characteristics covered by NICE and STOPS include shorter average lengths and the ability to distinguish between classes at a fine-granular level in NICE-45 and STOPS-41. The investigation of more documents is also conducted in the case of STOPS. As a result, NICE-45 and STOPS-41 reveal that DADGNN encounters issues when dealing with more classes, even falling around 20 and 60 percent points behind the baseline models. While still performing worse than the baseline models, InducT-GCN outperforms DADGNN on all four datasets. Transformers once again demonstrate their strength and rank as the top performing models across all datasets on this dataset. There are also significant drops. ERNIE 2.0 performs worse than the baseline models with 45.55% on NICE-45. However, ERNIE 2.0 (optimized), which uses different hyperparameters, comes in third with 67.65%

Table 3: Accuracy on our own short text classification datasets. The "Short Text" column indicates whether the model makes claims about its ability to categorize short texts. Provenance refers to the source of the accuracy scores.

Inductive Models	Short Text	NICE-45	NICE-2	STOPS-41	STOPS-2
Transformer Models					
BERT	N	**72.79**	99.72	89.4	99.87
RoBERTa	N	66.09	**99.76**	89.56	99.86
DeBERTa	N	59.42	99.72	**89.73**	99.85
ERNIE 2.0	N	45.55	99.69	89.39	99.85
ERNIE 2.0 (optimized)	N	67.65	99.72	89.65	**99.88**
DistilBERT	N	69.28	99.75	89.32	99.85
ALBERTv2	N	59.24	99.51	88.58	99.83
BoW Models					
WideMLP	N	58.99	96.76	88.2	97.05
Graph-based Models					
DADGNN	Y	28.51	91.15	26.75	97.48
InducT-GCN	N	47.31	94.97	86.11	97.71
CNN and LSTMs					
LSTM (BERT)	Y	47.81	96.63	86.27	96.05
Bi-LSTM (BERT)	Y	52.39	96.63	85.93	98.54
LSTM (GloVe)	Y	52.64	96.17	87.4	99.46
Bi-LSTM (GloVe)	Y	55.35	95.93	87.38	99.43

6 DISCUSSION

Graph-based models are computationally expensive because they require not only the creation of the graph but also its training, which can be resource- and time-intensive, especially for word-document graphs with $O(N^2)$ space [6]. On STOPS, this drawback becomes very apparent. We could observe that DADGNN required roughly 30 hours of training time, while BERT only took 30 minutes to fine-tune with the same resources. Although in the case of BERT, the pre-training was already very expensive, transfer learning allows this effort to be used for a variety of tasks. Nevertheless, the Transformers outperform the inductive graph-based models as well as the short text models, with just one exception. The best model for SearchSnippets is ESGNN, but additional preprocessing and a modified training split were employed. Our Bi-LSTM results, obtained without additional preprocessing, differ by 16.66 percentage points from the Bi-LSTM results from Zhao et al. [42]. This indicates that preprocessing, and not a better model, is primarily responsible for the strong outcomes of all of their SearchSnippets experiments. Another interesting discovery can be made using the sentiment datasets. In comparison to other datasets, Transformers outperforms graph-based models that do not utilize a Transformer themselves by a large margin. This demonstrates that graph-based models may not be as effective at sentiment prediction tasks. However, it should be noted that not all Transformers are consistently excellent. For instance, at NICE-45, one can observe a lower performance with ERNIE 2.0. But the absence of this performance decrease in our optimized version, ERNIE 2.0 (optimized), suggests that choosing suitable hyperparameters is crucial in this case.

6.1 Key Results

Our experiments unambiguously demonstrate that Transformers achieve SOTA accuracy on short text classification tasks. This raises the question of whether specialized short text techniques are necessary given that the performance of the existing models is insufficient. This observation is especially interesting because many of the short text models used are from 2021 [18, 32, 37, 42] or 2022 [29]. Most short text models attempt to enrich the documents with some kind of external context, such as a knowledge base or POS tags. However, one could argue that Transformers implicitly contain context in their weights through their pre-training.

Many of the short text models compare themselves to Transformers and assert that they outperform them. For instance, Ye et al. [39] claim to outperform BERT by 2.2 percentage points on MR, but their fine-tuned BERT only achieves 80.3%. In contrast, our own experiments show that BERT achieves 86.94%. With 85.86% on MR, Zhao et al. [42] achieve better BERT results, but only to beat it by a meager 0.2% with C-BERT. Given the low surplus, they would no longer outperform it with a marginally better selection of hyperparameters for BERT. Therefore, it is reasonable to assume that the importance of good hyperparameters for Transformers is underestimated and that they are often not properly optimized. ERNIE 2.0 (optimized), which outperforms ERNIE 2.0 on every dataset, also demonstrates the effect of better hyperparameters.

Additionally, there is a need for new short text datasets because the widely used benchmark datasets share many characteristics and fall short in many use cases. The common benchmark datasets all contain around 10, 000 documents, distinguish only a few classes, and frequently have a similar average length. Furthermore, many of them cover the same tasks. For instance, MR, Twitter, and SST-2 all perform sentiment prediction, which makes sense given how much short text is produced by social media. In this paper, we introduce two new datasets with distinctive attributes to cover more cases in NICE and STOPS. New and intriguing findings are produced by the new characteristics that are investigated using these datasets. Particularly, the ability to distinguish between classes at a fine-granular level reveals the shortcomings of various models, like DADGNN or ERNIE 2.0. NICE-45 in particular proved to be challenging for all models, making it a good benchmark for future advancements.

6.2 Threats to Validity

In our study, each experiment was conducted once, with the outcome being used as the final result. Many papers provide their results as the average of multiple runs. Nevertheless, the extremely low standard deviation [6, 18, 42] raises the prospect that this is unnecessary and does not justify the higher computational expense. We also acknowledge that STOPS contains user-generated labels, some of which may not be entirely accurate. However, given that this occurs frequently in numerous use cases, it is also crucial to test the models in these scenarios.

6.3 Generalization

As we cover in our experiments a range of diverse domains, with sentiment analysis on various themes (MR, SST-2, Twitter), question type classification (TREC), news (R8), and even search queries (SearchSnippets), we expect to find equivalent results on other short text classification datasets. Additionally, the categorization of goods and services is covered by our new datasets NICE and STOPS. They include additional features not covered by the benchmark datasets, including a significantly larger amount of training data in STOPS, a shorter average length, and the capacity to differentiate between a wider range of classes. By using an example from a business problem, STOPS specifically demonstrates how the knowledge gained here can be applied in corporate use.

In this work, we cover a variety of models for each architecture, particularly the most popular and best-performing ones. Our findings are consistent with the studies by Galke and Scherp [6], which demonstrate the tremendous power of Transformers for traditional text classification.

6.4 Future Work and Impact

Our research has a direct impact on the model selection for short text classification. For real-world application, in particular, further information is provided. Future research on even further improving the performance of Transformers on short text would be interesting. Pre-training on short texts or on in-domain texts (i. e., pre-training in the same domain as the target task) [2, 27, 30], multi-task fine-tuning [27, 30], or an ensemble of various Transformer models [43] may all be effective methods to improve performance. A brief study of the effects of a straightforward Transformer ensemble can be found in Appendix B.

7 CONCLUSION

Our experiments unequivocally demonstrate the outstanding capability of Transformers for short text classification tasks. Additional research on our newly released datasets, NICE and STOPS, supports these findings and highlights the issue of becoming overly dependent on benchmark datasets with a limited number of characteristics. In conclusion, our study raises the question of whether specialized short text techniques are required given the insufficient performance of current models.

For reproducibility, we make the source code available at https://github.com/FKarl/Bachelor-Thesis. This includes also the scripts to reproduce our two new datasets.

ACKNOWLEDGMENTS

We thank Till Blume and Felix Krieger from Ernst & Young (EY) for the discussion of the problem statement that motivated this work. We are grateful to Liu et al. [18] for providing the unreleased DADGNN source code.

REFERENCES

[1] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* 3 (2003), 993–1022. http://jmlr.org/papers/v3/blei03a.html

[2] Alexander Brinkmann and Christian Bizer. 2021. Improving Hierarchical Product Classification using Domain-specific Language Modelling. *IEEE Data Eng. Bull.* 44, 2 (2021), 14–25.

[3] Zhaoyang Deng, Chenxiang Sun, Guoqiang Zhong, and Yuxu Mao. 2022. Text Classification with Attention Gated Graph Neural Network. *Cognitive Computation* (2022), 1–10.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, 4171–4186. https://doi.org/10.18653/v1/n19-1423

[5] Kaize Ding, Jianling Wang, Jundong Li, Dingcheng Li, and Huan Liu. 2020. Be More with Less: Hypergraph Attention Networks for Inductive Text Classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 4927–4936. https://doi.org/10.18653/v1/2020.emnlp-main.399

[6] Lukas Galke and Ansgar Scherp. 2021. Bag-of-Words vs. Graph vs. Sequence in Text Classification: Questioning the Necessity of Text-Graphs and the Surprising Strength of a Wide MLP. *CoRR* abs/2109.03777 (2021). arXiv:2109.03777 https://arxiv.org/abs/2109.03777

[7] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: decoding-Enhanced Bert with Disentangled Attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. https://openreview.net/forum?id=XPZIaotutsD

[8] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[9] Yang Hu, Haoxuan You, Zhecan Wang, Zhicheng Wang, Erjin Zhou, and Yue Gao. 2021. Graph-MLP: node classification without message passing in graph. *arXiv preprint arXiv:2106.04051* (2021).

[10] Yen-Hao Huang, Yi-Hsin Chen, and Yi-Shin Chen. 2022. ConTextING: Granting Document-Wise Contextual Embeddings to Graph Neural Networks for Inductive Text Classification. In *Proceedings of the 29th International Conference on Computational Linguistics*. International Committee on Computational Linguistics, Gyeongju, Republic of Korea, 1163–1168. https://aclanthology.org/2022.coling-1.100

[11] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, 427–431. https://aclanthology.org/E17-2068

[12] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 1746–1751. https://doi.org/10.3115/v1/D14-1181

[13] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *CoRR* abs/1909.11942 (2019). arXiv:1909.11942 http://arxiv.org/abs/1909.11942

[14] Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S. Yu, and Lifang He. 2022. A Survey on Text Classification: From Traditional to Deep Learning. *ACM Trans. Intell. Syst. Technol.* 13, 2, Article 31 (apr 2022), 41 pages. https://doi.org/10.1145/3495162

[15] Xin Li and Dan Roth. 2002. Learning Question Classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*. https://aclanthology.org/C02-1150

[16] Yuxiao Lin, Yuxian Meng, Xiaofei Sun, Qinghong Han, Kun Kuang, Jiwei Li, and Fei Wu. 2021. BertGCN: Transductive Text Classification by Combining GNN and BERT. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Association for Computational Linguistics, Online, 1456–1462. https://doi.org/10.18653/v1/2021.findings-acl.126

[17] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent Neural Network for Text Classification with Multi-Task Learning. *CoRR* abs/1605.05101 (2016). arXiv:1605.05101 http://arxiv.org/abs/1605.05101

[18] Yonghao Liu, Renchu Guan, Fausto Giunchiglia, Yanchun Liang, and Xiaoyue Feng. 2021. Deep attention diffusion graph neural networks for text classification. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 8142–8152.

[19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692 (2019). arXiv:1907.11692 http://arxiv.org/abs/1907.11692

[20] Florian Mai, Lukas Galke, and Ansgar Scherp. 2018. Using Deep Learning For Title-Based Semantic Subject Indexing To Reach Competitive Performance to Full-Text. *CoRR* abs/1801.06717 (2018). arXiv:1801.06717 http://arxiv.org/abs/1801.06717

[21] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[22] Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. 2008. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of the 17th international conference on World Wide Web*. 91–100.

[23] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[24] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR* abs/1910.01108 (2019). arXiv:1910.01108 http://arxiv.org/abs/1910.01108

[25] Jinze Shi, Xiaoming Wu, Xiangzhi Liu, Wenpeng Lu, and Shu Li. 2022. Inductive Light Graph Convolution Network for Text Classification Based on Word-Label Graph. In *International Conference on Intelligent Information Processing*. Springer, 42–55.

[26] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher Manning, Andrew Ng, and Christopher Potts. 2013. Parsing With Compositional Vector Grammars. In *EMNLP*.

[27] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to Fine-Tune BERT for Text Classification?. In *Chinese Computational Linguistics - 18th China National Conference, CCL 2019, Kunming, China, October 18-20, 2019, Proceedings (Lecture Notes in Computer Science, Vol. 11856)*, Maosong Sun, Xuanjing Huang, Heng Ji, Zhiyuan Liu, and Yang Liu (Eds.). Springer, 194–206. https://doi.org/10.1007/978-3-030-32381-3_16

[28] Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2019. ERNIE 2.0: A Continual Pre-training Framework for Language Understanding. *CoRR* abs/1907.12412 (2019). arXiv:1907.12412 http://arxiv.org/abs/1907.12412

[29] Zhongtian Sun, Anoushka Harit, Alexandra I. Cristea, Jialin Yu, Lei Shi, and Noura Al Moubayed. 2022. Contrastive Learning with Heterogeneous Graph Attention Networks on Short Text Classification. In *2022 International Joint Conference on Neural Networks (IJCNN)*. 1–6. https://doi.org/10.1109/IJCNN55064.2022.9892257

[30] Lewis Tunstall, Leandro von Werra, and Thomas Wolf. 2022. *Natural language processing with Transformers*. O'Reilly Media, Inc.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[32] Haitao Wang, Keke Tian, Zhengjiang Wu, and Lei Wang. 2021. A short text classification method based on convolutional neural network and semantic extension. *International Journal of Computational Intelligence Systems* 14, 1 (2021), 367–375.

[33] Kunze Wang, Soyeon Caren Han, and Josiah Poon. 2022. InducT-GCN: Inductive Graph Convolutional Networks for Text Classification. *arXiv preprint arXiv:2206.00265* (2022).

[34] Yaqing Wang, Song Wang, Quanming Yao, and Dejing Dou. 2021. Hierarchical Heterogeneous Graph Representation Learning for Short Text Classification. *CoRR* abs/2111.00180 (2021). arXiv:2111.00180 https://arxiv.org/abs/2111.00180

[35] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Qili Zhu. 2012. Probase: a probabilistic taxonomy for text understanding. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*, K. Selçuk Candan, Yi Chen, Richard T. Snodgrass, Luis Gravano, and Ariel Fuxman (Eds.). ACM, 481–492. https://doi.org/10.1145/2213836.2213891

[36] Li Yang, Qifan Wang, Zac Yu, Anand Kulkarni, Sumit Sanghai, Bin Shu, Jon Elsas, and Bhargav Kanagal. 2022. Mave: A product dataset for multi-source attribute value extraction. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 1256–1265.

[37] Tianchi Yang, Linmei Hu, Chuan Shi, Houye Ji, Xiaoli Li, and Liqiang Nie. 2021. HGAT: Heterogeneous Graph Attention Networks for Semi-Supervised Short Text Classification. *ACM Trans. Inf. Syst.* 39, 3, Article 32 (may 2021), 29 pages. https://doi.org/10.1145/3450352

[38] Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph Convolutional Networks for Text Classification. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 01 (Jul. 2019), 7370–7377. https://doi.org/10.1609/aaai.v33i01.33017370

[39] Zhihao Ye, Gongyao Jiang, Ye Liu, Zhiyong Li, and Jin Yuan. 2020. Document and word representations generated by graph convolutional network and bert for short text classification. In *ECAI 2020*. IOS Press, 2275–2281.

[40] Fang Zeng, Niannian Chen, Dan Yang, and Zhigang Meng. 2022. Simplified-Boosting Ensemble Convolutional Network for Text Classification. *Neural Processing Letters* (2022), 1–16.

[41] Yufeng Zhang, Xueli Yu, Zeyu Cui, Shu Wu, Zhongzhen Wen, and Liang Wang. 2020. Every document owns its structure: Inductive text classification via graph neural networks. *arXiv preprint arXiv:2004.13826* (2020).

[42] Ke Zhao, Lan Huang, Rui Song, Qiang Shen, and Hao Xu. 2021. A Sequential Graph Neural Network for Short Text Classification. *Algorithms* 14, 12 (2021), 352.

[43] Honglei Zhuang, Zhen Qin, Shuguang Han, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2021. Ensemble Distillation for BERT-Based Ranking Models. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval* (Virtual Event, Canada) *(ICTIR '21)*. Association for Computing Machinery, New York, NY, USA, 131–136. https://doi.org/10.1145/3471158.3472238

Supplementary Materials

A HYPERPARAMETER OPTIMIZATION

For our own hyperparameter optimization, we used the BERT values from Galke and Scherp [6] as a starting point and then systematically tested different parameters for each model on every dataset. The hyperparameters are chosen using the Bayesian hyperparameter search method, which models the relationship between the parameters and accuracy using a Gaussian process. We chose intervals of the best-performing parameters retrieved from the Bayesian search method in order to obtain uniform parameters for all datasets. Our hyperparameters are then defined by the median of the intersection of these intervals. In our case, none of the intersections were empty.

B TRANSFORMER ENSEMBLE MODELS

In response to the improvement made by TextGCN-CNN-serial-SB from Zeng et al. [40], we implemented simplified ensemble models to exploit the concept in an inductive setup by utilizing the best models for each dataset. Along with the models that will be addressed and used in the experiment, we also tried a stacking strategy, using a meta-model to match the test data to the proper model, but this did not produce any improvements.

B.1 Ensemble Models

This section provides a detailed explanation of the simple ensemble models that we propose. A illustration of the models is shown in Figure 1 and Figure 2.

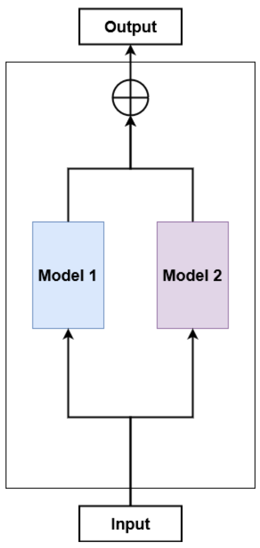

Figure 1: A representation of Top2Ensemble. $\bigoplus$ is the weighted sum operation from Equation 4.

B.1.1 Top2Ensemble. In Top2Ensemble, the best two Transformer models are trained independently using their respective parameters, and the results are then concatenated with a weighted sum. Formally, the concatenation is defined as:

$$predictions = \alpha \cdot m_1(\cdot) + (1 - \alpha) \cdot m_2(\cdot) \quad (4)$$

where m_1 is the best-performing Transformer model, and m_2 is the second-best Transformer model. An α value of 0.5 is selected for Top2Ensemble and is therefore interpretable as the mean.

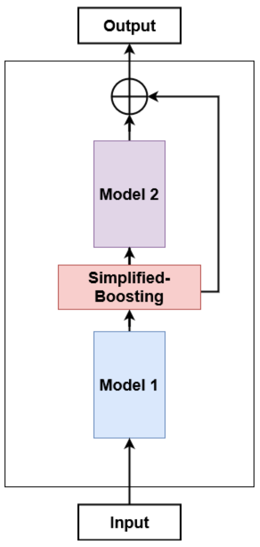

Figure 2: A representation of Top2Boost. $\bigoplus$ is the weighted sum operation from Equation 4.

B.1.2 Top2Boost. In Top2Boost, the second-best model is only trained on the best model's misclassifications with the goal of enhancing those during concatenation. The same formula as in Equation 4 is used for concatenation, but an α value of 0.7 is chosen due to the larger number of training samples for the best-performing model.

B.2 Results

The results for our ensemble methods are shown in Tables 4 and 5. Top2Ensemble was successful in enhancing the performance of the top model, while Top2Boost failed. The improvement in performance on SearchSnippets and STOPS-41 is negligible, but a 0.66 percent point improvement on SST-2 is notable. However, in the majority of cases, our simplified ensembles reduced the performance of the best model.

We examine the effects of various α in greater detail in Tables 6 and 7. The results support the choice of our α values of 0.5 and 0.7 because the best accuracy was obtained in the vicinity of these values.

Table 4: Accuracy on short text classification datasets using simple ensemble models

Transformer Ensemble Models	R8	MR	SearchSnippets	Twitter	TREC	SST-2
Top2Ensemble	98.36	89.79	89.78↑	99.97	99.20	95.44↑
Top2Boost	89.59	89.84	89.25	99.93	98.80	94.45

↑ Improvement in comparison to the best Transformer model

Table 5: Accuracy on our own short text classification datasets using simple ensemble models

Transformer Ensemble Models	NICE-45	NICE-2	STOPS-41	STOPS-2
Top2Ensemble	72.34	99.65	89.94↑	99.82
Top2Boost	72.20	99.72	88.89	99.78

↑ Improvement in comparison to the best Transformer model

Table 6: Impact of different α on Top2Ensemble

Dataset	R8	MR	SearchSnippets	Twitter	TREC	SST-2	NICE-45	NICE-2	STOPS-41	STOPS-2
Model 1	DeBERTa	DeBERTa	DistilBERT	ALBERTv2	BERT	DeBERTa	BERT	RoBERTa	DeBERTa	BERT
Model 2	RoBERTa	RoBERTa	ERNIE 2.0	ERNIE 2.0	DistilBERT	RoBERTa	DistilBERT	DistilBERT	RoBERTa	RoBERTa
$\alpha = 0.9$	98.26	89.65	89.34	99.93	99.00	94.67	71.20	**99.76**	89.69	99.79
$\alpha = 0.8$	98.31	89.79	89.43	99.93	99.00	94.95	71.37	**99.76**	89.83	99.79
$\alpha = 0.7$	**98.36**	90.01	89.56	**99.97**	**99.20**	95.22	71.37	99.69	89.91	99.80
$\alpha = 0.6$	**98.36**	**90.18**	89.52	**99.97**	**99.20**	95.39	72.03	99.69	**89.96**	99.81
$\alpha = 0.5$	**98.36**	89.79	**89.78**	**99.97**	**99.20**	**95.44**	**72.34**	99.65	89.94	**99.82**
$\alpha = 0.4$	98.22	89.50	89.30	**99.97**	99.00	95.11	72.06	99.65	89.92	99.81
$\alpha = 0.3$	98.08	89.39	89.47	**99.97**	99.00	95.12	71.51	99.72	89.92	99.81
$\alpha = 0.2$	98.04	89.11	89.17	**99.97**	99.00	94.95	71.20	99.62	89.78	99.79
$\alpha = 0.1$	97.99	88.80	89.04	**99.97**	99.00	94.95	70.40	99.58	89.63	99.77

Table 7: Impact of different α on Top2Boost

Dataset	R8	MR	SearchSnippets	Twitter	TREC	SST-2	NICE-45	NICE-2	STOPS-41	STOPS-2
Model 1	DeBERTa	DeBERTa	DistilBERT	ALBERTv2	BERT	DeBERTa	BERT	RoBERTa	DeBERTa	BERT
Model 2	RoBERTa	RoBERTa	ERNIE 2.0	ERNIE 2.0	DistilBERT	RoBERTa	DistilBERT	DistilBERT	RoBERTa	RoBERTa
$\alpha = 0.9$	89.65	**89.90**	89.12	**99.93**	**98.80**	**94.45**	71.09	99.69	**89.50**	**99.78**
$\alpha = 0.8$	89.65	89.87	**89.25**	**99.93**	**98.80**	**94.45**	71.89	99.69	89.42	**99.78**
$\alpha = 0.7$	89.59	89.84	**89.25**	**99.93**	**98.80**	**94.45**	**72.20**	**99.72**	88.89	**99.78**
$\alpha = 0.6$	89.59	89.84	88.86	**99.93**	**98.80**	94.40	71.96	**99.72**	87.26	**99.78**
$\alpha = 0.5$	89.62	89.87	88.51	**99.93**	**98.80**	94.34	71.37	**99.72**	84.69	**99.78**
$\alpha = 0.4$	89.62	89.84	88.25	**99.93**	98.00	94.29	69.32	99.62	81.25	**99.78**
$\alpha = 0.3$	89.62	89.81	87.54	**99.93**	87.60	94.12	66.54	99.51	77.61	**99.78**
$\alpha = 0.2$	89.67	89.67	82.81	**99.93**	84.20	93.90	64.32	98.96	73.68	**99.78**
$\alpha = 0.1$	**89.70**	89.45	67.23	68.36	65.00	92.92	60.42	86.97	70.06	98.43