Bag-of-Words vs. Graph vs. Sequence in Text Classification: Questioning the Necessity of Text-Graphs and the Surprising Strength of a Wide MLP

Lukas Galke
University of Kiel, Germany
MPI for Psycholinguistics, the Netherlands
`lukas.galke@mpi.nl`

Ansgar Scherp
University of Ulm, Germany
`ansgar.scherp@uni-ulm.de`

Abstract

Graph neural networks have triggered a resurgence of graph-based text classification methods, defining today's state of the art. We show that a wide multi-layer perceptron (MLP) using a Bag-of-Words (BoW) outperforms the recent graph-based models TextGCN and Hete-GCN in an inductive text classification setting and is comparable with HyperGAT. Moreover, we fine-tune a sequence-based BERT and a lightweight DistilBERT model, which both outperform all state-of-the-art models. These results question the importance of synthetic graphs used in modern text classifiers. In terms of efficiency, DistilBERT is still twice as large as our BoW-based wide MLP, while graph-based models like TextGCN require setting up an $\mathcal{O}(N^2)$ graph, where N is the vocabulary plus corpus size. Finally, since Transformers need to compute $\mathcal{O}(L^2)$ attention weights with sequence length L, the MLP models show higher training and inference speeds on datasets with long sequences.

1 Introduction

Text categorization is the task of assigning topical categories to text units such as documents, social media postings, or news articles. Research on text categorization is a very active field as just the sheer amount of new methods in recent surveys shows (Bayer et al., 2021; Li et al., 2020; Zhou et al., 2020; Kowsari et al., 2019; Kadhim, 2019).

There are approaches based on a Bag of Words (BoW) that perform text categorization purely on the basis of a multiset of tokens. Among them are Deep Averaging Networks (DAN) (Iyyer et al., 2015), a deep Multi-Layer Perceptron (MLP) model with n layers that relies on averaging the BoW, Simple Word Embedding Models (SWEM) (Shen et al., 2018) that explores different pooling strategies for pretrained word embeddings, and fastText (Bojanowski et al., 2017), which uses a linear layer on top of pretrained word embed-

dings. These models count the occurrence of all tokens in the input sequence, while disregarding word position and order, and then rely on word embeddings and fully connected feedforward layer(s). We call these **BoW-based models**.

Among the most popular recent methods for text categorization are graph-based models such as TextGCN (Yao et al., 2019) that first induce a synthetic word-document co-occurence graph over the corpus and subsequently apply a graph neural network (GNN) to perform the classification task. Besides TextGCN, there are follow-up works like HeteGCN (Ragesh et al., 2021), TensorGCN (Liu et al., 2020), and HyperGAT (Ding et al., 2020), which we collectively call **graph-based models**.

Finally, there is the well-known Transformer (Vaswani et al., 2017) universe with models such as BERT (Devlin et al., 2019) and its size-reduced variants such as DistilBERT (Sanh et al., 2019). Here, the input is a (fixed-length) sequence of tokens, which is then fed into multiple layers of self-attention. Lightweight versions such as Distil-BERT and others (Tay et al., 2020; Fournier et al., 2021) use less parameters but operate on the same type of input. Together with recurrent models such as LSTMs, we call these **sequence-based models**.

In this paper, we hypothesize that text categorization can be very well conducted by simple but effective BoW-based models. We investigate this research question in three steps: First, we conduct an in-depth analysis of the literature. We review the key research in the field of text categorization. From this analysis, we derive the different families of methods, the established benchmark datasets, and identify the top performing methods. We decide for which models we report numbers from the literature and which models we run on our own. Overall, we compare 16 different methods from the families of BoW-based models (8 methods), sequence-based models (3 methods), and graph-based models (5 methods). We run our own experi-

ments for 7 of these methods on 5 text categorization datasets, while we report the results from the literature for the remaining methods.

The result is surprising: Our own BoW-based MLP, called the WideMLP, with only one wide hidden layer, outperforms many of the recent graph-based models for inductive text categorization (Yao et al., 2019; Liu et al., 2020; Ragesh et al., 2021). Moreover, we did not find any reported scores for BERT-based methods from the sequence-based family. Thus, we fine-tuned our own BERT (Devlin et al., 2019) and DistilBERT (Sanh et al., 2019). These models set a new state of the art. On a meta-level, our study shows that MLPs have largely been ignored as competitor methods in experiments. It seems as if MLPs have been forgotten as baseline in the literature, which instead is focusing mostly on other advanced Deep Learning architectures. Considering strong baselines is, however, an important means to argue about *true* scientific advancement (Shen et al., 2018; Dacrema et al., 2019). Simple models are also often preferred in industry due to lower operational and maintenance costs.

Below, we introduce our methodology and results from the literature study. Subsequently, we introduce the families of models in Section 3. Thereafter, we describe the experimental procedure in Section 4. We present the results of our experiments in Section 5 and discuss our findings in Section 6, before we conclude.

2 Literature on Text Categorization

Methodology In a first step, we have analyzed recent surveys on text categorization and comparison studies (Minaee et al., 2021; Bayer et al., 2021; Li et al., 2020; Zhou et al., 2020; Kowsari et al., 2019; Kadhim, 2019; Galke et al., 2017; Zhang et al., 2016). These cover the range from shallow to deep classification models. Second, we have screened for literature in key NLP and AI venues. Finally, we have complemented our search by checking results and papers on paperswithcode.com. On the basis of this input, we have determined three families of methods and benchmark datasets (see Table 2). We focus our analysis on identifying models per family showing strong performance and select the methods to include in our study. For all models, we have verified that the same train-test split is used. We check whether modified versions of the datasets have been used (e. g., fewer classes), to avoid bias and wrongfully giving advantages.

BoW-based Models Classical machine learning models that operate on a BoW-based input are extensively discussed in two surveys (Kowsari et al., 2019; Kadhim, 2019) and other comparison studies (Galke et al., 2017). Iyyer et al. (2015) proposed DAN, which combine word embeddings and deep feedforward networks. It is an MLP with 1-6 hidden layers, non-linear activation, dropout, and Ada-Grad as optimization method. The results suggest to use pretrained embeddings such as GloVe (Pennington et al., 2014) over a randomly initialized neural bag of-words (Kalchbrenner et al., 2014) as input. In fastText (Bojanowski et al., 2017; Joulin et al., 2017) a linear layer on top of pretrained embeddings is used for classification. Furthermore, Shen et al. (2018) explore embedding pooling variants and find that SWEM can rival approaches based on recurrent (RNN) and convolutional neural networks (CNN). We consider fastText, SWEM, and a DAN-like deeper MLP in our comparison.

Note that those approaches that rely on logistic regression on top of pretrained word embeddings, e. g., fastText, share a similar architecture as an MLP with one hidden layer. However, the standard training protocol involves pretraining the word embedding on large amounts of unlabeled text and then freezing the word embeddings while training the logistic regression (Mikolov et al., 2013).

Graph-based Models Using graphs induced from text for the task of text categorization has a long history in the community. An early work is the term co-occurrence graph of the KeyGraph algorithm (Ohsawa et al., 1998). The graph is split into segments, representing the key concepts in the document. Co-occurence graphs have also been used for automatic keyword extraction such as in RAKE (Rose et al., 2010) and can be also used for classification (Zhang et al., 2021).

Modern approaches exploit this idea in combination with graph neural networks (GNN) (Hamilton, 2020). Examples of GNN-based methods operating on a word-document co-occurence graph are TextGCN (Yao et al., 2019) and its successor TensorGCN (Liu et al., 2020) as well as Hete-GCN (Ragesh et al., 2021), HyperGAT (Ding et al., 2020), and DADGNN (Liu et al., 2020). We briefly discuss these models: In TextGCN, the authors set up a graph based on word-word connections given by window-based pointwise mutual information and word-document TF-IDF scores. They use a one-hot encoding as node features and apply a

two-layer graph convolutional network (Kipf and Welling, 2017) on the graph to carry out the node classification task. HeteGCN combines ideas from Predictive Text Embedding (Tang et al., 2015) and TextGCN and split the adjacency matrix into its word-document and word-word sub-matrices and fuse the different layers' representations when required. TensorGCN uses multiple ways of converting text data into graph data including a semantic graph created with an LSTM, a syntactic graph created by dependency parsing, and a sequential graph based on word co-occurrence. HyperGAT extended the idea of text-induced graphs for text classification to hypergraphs. The model uses graph attention and two kinds of hyperedges. Sequential hyperedges represent the relation between sentences and their words. Semantic hyperedges for word-word connections are derived from topic models (Blei et al., 2001). Finally, DADGNN is a graph-based approach that uses attention diffusion and decoupling techniques to tackle oversmoothing of the GNN and to be able to stack more layers.

In TextGCN's original transductive formulation, the entire graph including the test set needs to be known for training. This may be prohibitive in practical applications as each batch of new documents would require retraining the model. When these methods are adapted for inductive learning, where the test set is unseen, they achieve notably lower scores (Ragesh et al., 2021). GNNs for text classification use corpus statistics, e. g., pointwise mutual information (PMI), to connect related words in a graph (Yao et al., 2019). When these were omitted, the GNNs would collapse to bag-of-words MLPs. Thus, GNNs have access to more information than BoW-MLPs. GloVe (Pennington et al., 2014) also captures PMI corpus statistics, which is why we include an MLP on GloVe input representations.

Sequence models: RNN and CNN   Recurrent neural networks (RNN) are a natural choice for any NLP task. However, it turned out to be challenging to find numbers reported on text categorization in the literature that can be used as references. The bidirectional LSTM with two-dimensional max pooling BLSTM-2DCNN (Zhou et al., 2016) has been applied on a stripped-down to 4 classes version of the 20ng dataset. Thus, the high score of 96.5 reported for 4ng cannot be compared with papers applied on the full 20ng dataset. Also Text-RCNN (Lai et al., 2015), a model combining recurrence and convolution uses only the 4 major

categories in the 20ng dataset. The results of Text-RCNN are identical with BLSTM-2DCNN. For the MR dataset, BLSTM-2DCNN provides no information on the specific split of the dataset. RNN-Capsule (Wang et al., 2018) is a sentiment analysis method reaching an accuracy of 83.8 on the MR dataset, but with a different train-test split. Lyu and Liu (2020) combine a 2D-CNN with bidirectional RNN. Another work applying a combination of a convolutional layer and an LSTM layer is by Wang et al. (2019b). The authors experiment with five English and two Chinese datasets, which are not in the set of representative datasets we identified. The authors report that their approach outperforms existing models like fastText on two of the five English datasets and both Chinese datasets.

Sequence models: Transformers   Surprisingly, only few works consider Transformer models for text categorization.   A recent work shows that BERT outperforms classic TF-IDF BoW approaches on English, Chinese, and Portuguese text classification datasets (González-Carvajal and Garrido-Merchán, 2020). We have not found any results of transformer-based models reported on those text categorization datasets that are commonly used in the graph-based approaches.

Therefore, we fine-tune BERT (Devlin et al., 2019) and DistilBERT (Sanh et al., 2019) on those datasets ourselves. BERT is a large pretrained language model on the basis of Transformers. DistilBERT (Sanh et al., 2019) is a distilled version of BERT with 40% reduced parameters while retaining 97% of BERT's language understanding capabilities. TinyBERT (Jiao et al., 2020) and MobileBERT (Sun et al., 2020) would be similarly suitable alternatives, among others. We chose DistilBERT because it can be fine-tuned independently from the BERT teacher. Its inference times are 60% faster than BERT, which makes it more likely to be reusable by labs with limited resources.

Summary   From our literature survey, we see that all recent methods are based on graphs. BoW-based methods are hardly found in experiments, while, likewise surprisingly, Transformer-based sequence models are extremely scarce in the literature on topical text categorization. The recent surveys on text categorization include both classical and Deep Learning models, but none considered a simple MLP except for the inclusion of DAN (Iyyer et al., 2015) in Li et al. (2020).

Model	Synthetic Graph	Position-Aware	Arbitrary Length	Inductive
Bag-of-Words	No	No	Yes	Yes
Graph: TextGCN	Yes	No	Yes	No
Graph: TensorGCN	Yes	Yes	Yes	No
Graph: HeteGCN/HyperGAT	Yes	No	Yes	Yes
Sequence: RNN/CNN	No	Yes	Yes	Yes
Sequence: BERT/DistilBERT	No	Yes	No	Yes

Table 1: Properties of text categorization approaches. Graph-based models that rely on having access to unlabeled test documents such as TextGCN and TensorGCN are not capable of inductive learning without modifications.

3   Models for Text Categorization

We formally introduce the three families of models for text categorization, namely the BoW-based, graph-based, and sequence-based models. Table 1 summarizes the key properties of the approaches: whether they require a synthetic graph, whether word position is reflected in the model, whether the model can deal with arbitrary length text, and whether the model is capable of inductive learning.

3.1   BoW-Based Text Categorization

Under pure BoW-based text categorization, we denote approaches that are not order-aware and operate only on the multiset of words from the input document. Given paired training examples $(\boldsymbol{x}, y) \in \mathcal{D}$, each consisting of a bag-of-words $\boldsymbol{x} \in \mathbb{R}^{n_{\mathrm{vocab}}}$ and a class label $y \in \mathbb{Y}$, the goal is to learn a generalizable function $\hat{\boldsymbol{y}} = f_\theta^{(\mathrm{BoW})}(\boldsymbol{x})$ with parameters θ such that $\arg\max(\hat{\boldsymbol{y}})$ preferably equals the true label y for input $\boldsymbol{x}$.

As BoW-based model, we consider a one hidden layer WideMLP (i. e., two layers in total). We experiment with pure BoW, TF-IDF weighted, and averaged GloVe input representations. We also use a two hidden layers WideMLP-2. We list the numbers for fastText, SWEM, and logistic regression from Ding et al. (2020) in our comparison.

3.2   Graph-Based Text Categorization

Graph-based text categorization approaches first set up a *synthetic* graph on the basis of the text corpus $\mathcal{D}$ in the form of an adjacency matrix $\hat{\boldsymbol{A}} := \mathrm{make\text{-}graph}(\mathcal{D})$. For instance, in TextGCN the graph is set up in two parts: word-word connections are modeled by pointwise mutual information and word-document edges resemble that the word occurs in the document. Then, a parameterized function $f_\theta^{(\mathrm{graph})}(\boldsymbol{X}, \hat{\boldsymbol{A}})$ is learned that uses the graph as input, where $\boldsymbol{X}$ are the node features. The graph is composed of word and document nodes, each receiving its own embedding (by setting $\boldsymbol{X} = \boldsymbol{I}$). In inductive learning, however, there is no embedding of the test documents. Note that the graph-based approaches from the current literature such as TextGCN also disregard word order, similar to the BoW-based models described above. A detailed discussion of the connection between TextGCN and MLP is provided in Appendix B.

We consider top performing graph-based models from the literature, namely TextGCN along with its successors HeteGCN, TensorGCN, HyperGAT, DADGNN, as well as simplified GCN (SGC) (Wu et al., 2019). We do not run our own experiments for the graph-based models but rely on the original work and extensive studies by Ding et al. (2020) and Ragesh et al. (2021).

3.3   Sequence-Based Text Categorization

We consider RNNs, LSTMs, and Transformers as sequence-based models. These models are aware of the order of the words in the input text in the sense that they are able to exploit word order information. Thus, the key difference to the BoW-based and graph-based families is that the word order is reflected by sequence-based model. The model signature is $\hat{y} = f_\theta^{(\mathrm{sequence})}(\langle x_1, x_2, \ldots, x_k \rangle)$, where k is the (maximum) sequence length. Word position is modeled by a dedicated positional encoding. For instance, in BERT each position is associated with an embedding vector that is added to the word embedding at input level.

For the sequence-based models, we run our own experiments with BERT and DistilBERT, while reporting the scores of a pretrained LSTM from Ding et al. (2020) for comparison.

4   Experimental Apparatus

4.1   Datasets

We use the same datasets and train-test split as in TextGCN (Yao et al., 2019). Those datasets are 20ng, R8, R52, ohsumed, and MR. Twenty

Newsgroups (20ng)[1] (bydate version) contains long posts categorized into 20 newsgroups. The mean sequence length is 551 words with a standard deviation (SD) of 2,047. R8 and R52 are subsets of the Reuters 21578 news dataset with 8 and 52 classes, respectively. The mean sequence length and SD is 119 ± 128 words for R8, and 126 ± 133 words for R52. Ohsumed[2] is a corpus of medical abstracts from the MEDLINE database that are categorized into diseases (one per abstract). The mean sequence length is 285 ± 123 words. Movie Reviews (MR)[3] (Pang and Lee, 2005), split by Tang et al. (2015), is a binary sentiment analysis dataset on sentence level (mean sequence length and SD: 25 ± 11). Table 2 shows the dataset characteristics.

Table 2: Characteristics of text classification datasets

Dataset	N	#Train	#Test	#Classes
20ng	18,846	11,314	7,532	20
R8	7,674	5,485	2,189	8
R52	9,100	6,532	2,568	52
ohsumed	7,400	3,357	4,043	23
MR	10,662	7,108	3,554	2

4.2 Inductive and Transductive Setups

We distinguish between a transductive and an inductive setup for text categorization. In the transductive setup, as used in TextGCN, the test documents are visible and actually used for the preprocessing step. In the inductive setting, the test documents remain unseen until test time (i. e., they are not available for preprocessing). We report the scores of the graph-based models for both setups from the literature, where available. BoW-based and sequence-based models are inherently inductive. Ragesh et al. (2021) have evaluated a variant of TextGCN that is capable of inductive learning, which we include in our results, too.

4.3 Procedure and Hyperparameter Settings

We have extracted accuracy scores from the literature according to our systematic selection from Section 2. Below, we provide a detailed description of the procedure for the models that we have run ourselves. We borrow the tokenization strategy

---

[1] http://qwone.com/~jason/20Newsgroups/
[2] http://disi.unitn.it/moschitti/corpora.htm
[3] https://www.cs.cornell.edu/people/pabo/movie-review-data/

from BERT (Devlin et al., 2019) along with its uncased vocabulary. The tokenizer relies primarily on WordPiece (Wu et al., 2016) for a high coverage while maintaining a small vocabulary.

Training our BoW-Models. Our WideMLP has one hidden layer with 1,024 rectified linear units (one input-to-hidden and one hidden-to-output layer). We apply dropout after each hidden layer, notably also after the initial embedding layer. Only for GloVe+WideMLP, neither dropout nor ReLU is applied to the *frozen* pretrained embeddings but only on subsequent layers. The variant WideMLP-2 has two ReLU-activated hidden layers (three layers in total) with $1,024$ hidden units each. While this might be overparameterized for single-label text classification tasks with few classes, we rely on recent findings that overparameterization leads to better generalization (Neyshabur et al., 2018; Nakkiran et al., 2020). In pre-experiments, we realized that MLPs are not very sensitive to hyperparameter choices. Therefore, we optimize cross-entropy with Adam (Kingma and Ba, 2015) and its default learning rate of 10^{-3}, a linearly decaying learning rate schedule and train for a high amount of steps (Nakkiran et al., 2020) (we use 100 epochs) with small batch sizes (we use 16) for sufficient stochasticity, along with a dropout ratio of 0.5.

Fine-tuning our BERT models. For BERT and DistilBERT, we fine-tune for 10 epochs with a linearly decaying learning rate of $5 \cdot 10^{-5}$ and an effective batch size of 128 via gradient accumulation of 8 x 16 batches. We truncate all inputs to 512 tokens. To isolate the influence of word order on BERT's performance, we conduct two further ablations. First, we set all position embeddings to zero and disable their gradient (**BERT w/o pos ids**). By doing this, we force BERT to operate on a bag-of-words without any notion of word order or position. Second, we shuffle each sequence to augment the training data. We use this augmentation strategy to increase the number of training examples by a factor of two (**BERT w/ shuf. augm.**).

4.4 Measures

We report accuracy as evaluation metric, which is equivalent to Micro-F1 in single-label classification (see Appendix C). We repeat all experiments five times with different random initialization of the parameters and report the mean and standard deviation of these five runs.

5 Results

5.1 Effectiveness

Table 3 shows the accuracy scores for the text categorization models on the five datasets. All graph-based models in the transductive setting show similar accuracy scores (maximum difference is 2 points). As expected, the scores decrease in the inductive setting up to a point where they are matched or even outperformed by our WideMLP.

In the inductive setting, the WideMLP models perform best among the BoW models, in particular, TFIDF+WideMLP and WideMLP on an unweighted BoW. The best-performing graph-based model is HyperGAT, yet DADGNN has a slight advantage on R8, R52, and MR. For the sequence-based models, BERT attains the highest scores, closely followed by DistilBERT.

The strong performance of WideMLP rivals all graph-based techniques reported in the literature, in particular, the recently published graph-inducing methods. MLP only falls behind HyperGAT, which relies on topic models to set up the graph. Another observation is that 1 hidden layer (but wide) is sufficient for the tasks, as the scores for MLP variants with 2 hidden layers are lower. We further observe that both pure BoW and TF-IDF weighted BoW lead to better results than approaches that exploit pretrained word embeddings such as GloVe-MLP, fastText, and SWEM.

With its immense pretraining, BERT yields the overall highest scores, closely followed by Distil-BERT. DistilBERT outperforms HyperGAT by 7 points on the MR dataset while being on par on the others. BERT outperforms the strongest graph-based competitor, HyperGAT, by 8 points on MR, 1.5 points on ohsumed, 1 point on R52 and R8, and 0.5 points on 20ng.

Our results further confirm that position embeddings are important for BERT with a notable decrease when those are omitted. Augmenting the data with shuffled sequences has led to neither a consistent decrease nor increase in performance.

5.2 Efficiency

Parameter Count of the Models Table 4 lists the parameter counts of the models. Even though the MLP is fully-connected on top of a bag-of-words with the dimensionality of the vocabulary size, it has only half of the parameters as Distil-BERT and a quarter of the parameters of BERT. Using TF-IDF does not change the number of model parameters. Due to the high vocabulary size, GloVe-based models have a high number of parameters, but the majority of those is frozen, i. e., does not get gradient updates during training.

Runtime Performance of the Models We provide the *total* running times in Table 5 as observed while conducting the experiments on a single NVIDIA A100-SXM4-40GB card. All WideMLP variants are an order of magnitude faster than DistilBERT when considering the average runtime *per epoch*. DistilBERT is twice as fast as the original BERT. The transformers are only faster than BoW models on the MR dataset. This is because the sequences in the MR dataset are much shorter and less $\mathcal{O}(L^2)$ attention weights have to be computed.

6 Discussion

Key Insights Our experiments show that our MLP models using BoW outperform the recent graph-based models TextGCN and HeteGCN in an inductive text classification setting. Furthermore, the MLP models are comparable to Hyper-GAT. Only transformer-based BERT and Distil-BERT models outperform our MLP and set a new state-of-the-art. This result is important for two reasons: First, the strong performance of a pure BoW-MLP questions the added value of synthetic graphs in models like TextGCN to the text categorization task. Only HyperGAT, which uses the expensive Latent Dirichlet Allocation for computing the graph, slightly outperforms our BoW-WideMLP in two out of five datasets. Thus, we argue that using strong baseline models for text classification is important to assess the true scientific advancement (Dacrema et al., 2019).

Second, in contrast to conventional wisdom (Iyyer et al., 2015), we find that pretrained embeddings, e. g., GloVe, can have a detrimental effect when compared to using an MLP with one wide hidden layer. Such an MLP circumvents the bottleneck of the small dimensionality of word embeddings and has a higher capacity. Furthermore, we experiment with more hidden layers (see WideMLP-2), but do not observe any improvement when the single hidden layer is sufficiently wide. A possible explanation is that already a single hidden layer is sufficient to approximate any compact function to an arbitrary degree of accuracy depending on the width of the hidden layer (Cybenko, 1989).

Finally, a new state-of-the-art is set by the transformer model BERT, which is not very surpris-

Table 3: Accuracy and standard deviation on text classification datasets. Column "Provenance" reports the source.

Inductive Setting	20ng	R8	R52	ohsumed	MR	Provenance
BoW-Models						
Log. Regression	83.70	93.33	90.65	61.14	76.28	Ragesh et al. (2021)
SWEM	85.16 (0.29)	95.32 (0.26)	92.94 (0.24)	63.12 (0.55)	76.65 (0.63)	Ding et al. (2020)
fastText	79.38 (0.30)	96.13 (0.21)	92.81 (0.09)	57.70 (0.49)	75.14 (0.20)	Ding et al. (2020)
TF-IDF + WideMLP	84.20 (0.16)	97.08 (0.16)	93.67 (0.23)	66.06 (0.29)	76.32 (0.17)	own experiment
WideMLP	83.31 (0.22)	97.27 (0.12)	93.89 (0.16)	63.95 (0.13)	76.72 (0.26)	own experiment
WideMLP-2	81.02 (0.23)	96.61 (1.22)	93.98 (0.23)	61.71 (0.33)	75.91 (0.51)	own experiment
GloVe+WideMLP	76.80 (0.11)	96.44 (0.08)	93.58 (0.06)	61.36 (0.22)	75.96 (0.17)	own experiment
GloVe+WideMLP-2	76.33 (0.18)	96.50 (0.14)	93.19 (0.11)	61.65 (0.27)	75.72 (0.45)	own experiment
Graph-based Models						
TextGCN	80.88 (0.54)	94.00 (0.40)	89.39 (0.38)	56.32 (1.36)	74.60 (0.43)	Ragesh et al. (2021)
HeteGCN	84.59 (0.14)	97.17 (0.33)	93.89 (0.45)	63.79 (0.80)	75.62 (0.26)	Ragesh et al. (2021)
HyperGAT	86.62 (0.16)	97.07 (0.23)	94.98 (0.27)	69.90 (0.34)	78.32 (0.27)	Ragesh et al. (2021)
DADGNN	—	98.15 (0.16)	95.16 (0.22)	—	78.64 (0.29)	Liu et al. (2021b)
Seq.-based Models						
LSTM (pretrain)	75.43 (1.72)	96.09 (0.19)	90.48 (0.86)	51.10 (1.50)	77.33 (0.89)	Ding et al. (2020)
DistilBERT	86.24 (0.26)	97.89 (0.15)	95.34 (0.08)	69.08 (0.60)	85.10 (0.33)	own experiment
BERT	87.21 (0.18)	98.03 (0.24)	96.17 (0.33)	71.46 (0.54)	86.61 (0.38)	own experiment
BERT w/o pos emb	81.47 (0.49)	97.39 (0.20)	94.70 (0.27)	65.18 (1.53)	80.35 (0.20)	own experiment
BERT w/ shuf. augm.	86.46 (0.42)	98.07 (0.21)	96.48 (0.18)	70.94 (0.60)	86.23 (0.33)	own experiment
Transductive Setting	**20ng**	**R8**	**R52**	**ohsumed**	**MR**	**Provenance**
Graph-based Models						
TextGCN	86.34	97.07	93.56	68.36	76.74	Yao et al. (2019)
SGC	88.5 (0.1)	97.2 (0.1)	94.0 (0.2)	68.5 (0.3)	75.9 (0.3)	Wu et al. (2019)
TensorGCN	87.74	98.04	95.05	70.11	77.91	Liu et al. (2020)
HeteGCN	87.15 (0.15)	97.24 (0.51)	94.35 (0.25)	68.11 (0.70)	76.71 (0.33)	Ragesh et al. (2021)

Table 4: Parameter counts of the models

Model	#parameters
WideMLP	31.3M
WideMLP-2	32.3M
GloVe+WideMLP	575,2M (frozen) + 0.3M
GloVe+WideMLP-2	575,2M (frozen) + 1.3M
DistilBERT	66M
BERT	110M

ing. However, as our efficiency analysis shows, the MLPs require only a fraction of the parameters and are faster in their combined training and inference time except for the MR dataset. The attention mechanism of (standard) Transformers is quadratic in the sequence length, which leads to slower processing of long sequences. With larger batches, the speed of the MLP could be increased even further.

Detailed Discussion of Results Graph-based models come with high training costs, as not only the graph has to be first computed, but also a GNN has to be trained. For standard GNN methods, the whole graph has to fit into the GPU memory and mini-batching is nontrivial, but possible with dedicated sampling techniques for GNNs (Fey et al., 2021). Furthermore, the original TextGCN is inher-

ently transductive, i. e., it has to be retrained whenever new documents appear. Strictly transductive models are effectively useless in practice (Lu et al., 2019) except for applications, in which a partially labeled corpus needs to be fully annotated. However, recent extensions such as HeteGCN, Hyper-GAT, and DADGNN already relax this constraint and enable inductive learning. Nevertheless, word-document graphs require $\mathcal{O}(N^2)$ space, where N is the number of documents plus the vocabulary size, which is a hurdle for large-scale applications.

There are also tasks where the *natural* structure of the graph data provides more information than the mere text, e. g., citations networks or connections in social graphs. In such cases, the performance of graph neural networks is the state of the art (Kipf and Welling, 2017; Velickovic et al., 2018) and are superior to MLPs that use only the node features and not the graph structure (Shchur et al., 2018). GNNs also find application in various NLP tasks, other than classification (Wu et al., 2021).

An interesting factor is the ability of the models to capture word order. BoW models disregard word order entirely and yield good results, but still fall behind order-aware Transformer models. In an extensive study, Conneau et al. (2018) have shown

Table 5: Total runtime (training+inference). Average of five runs rounded to minutes.

Model	#epochs	20ng	R8	R52	ohsumed	MR
WideMLP	100	7min	3min	4min	3min	4min
TF-IDF+WideMLP	100	9min	4min	4min	3min	4min
WideMLP-2	100	9min	5min	5min	3min	6min
GloVe+WideMLP	100	6min	3min	4min	3min	4min
GloVe+WideMLP-2	100	6min	4min	4min	3min	4min
DistilBERT	10	8min	4min	5min	3min	1min
BERT	10	15min	7min	8min	5min	2min

that memorizing the word content (which words appear at all) is most indicative of downstream task performance. Sinha et al. (2021) have experimented with pretraining BERT by disabling word order during pretraining and show that it makes surprisingly little difference for fine-tuning. In their study, word order is preserved during fine-tuning. In our experiments, we have conducted complementary experiments: we have used a BERT model that is pretrained with word order, but we have deactivated the position encoding during fine-tuning. Our results show that there is a notable drop in performance but the model does not fail completely.

Other NLP tasks such as question answering (Rajpurkar et al., 2016) or natural language inference (Wang et al., 2019a) can also be regarded as text classification on a technical level. Here, the positional information of the sequence is more important than for pure topical text categorization. One can expect that BoW-based models perform worse than sequence-based models.

Generalizability We expect that similar observations would be made on other text classification datasets because we have already covered a range of different characteristics: long and short texts, topical categorization (20ng, Reuters, and Ohsumed) and sentiment prediction (MR) in the domains of forum postings, news, movie reviews, and medical abstracts. Our results are in line with those from other fields, who have reported a resurgence of MLPs. For example, in business prediction, an MLP baseline outperforms various other Deep Learning models (Venugopal et al., 2021; Yedida et al., 2021). In computer vision, Tolstikhin et al. (2021) and Melas-Kyriazi (2021) proposed attention-free MLP models that are on par with the Vision Transformer (Dosovitskiy et al., 2021). In natural language processing, Liu et al. (2021a) show similar results, while acknowledging that a small attention module is necessary for some tasks.

Threats to Validity We acknowledge that the experimental datasets are limited to English. While word order is important in the English language, it is notable that methods that discard word order still work well for text categorization. Another possible bias is the comparability of the results. However, we carefully checked all relevant parameters such as the train/test split, the number of classes in the datasets, if datasets have been pre-processed in such a way that, e. g., makes a task easier like reducing the number of classes, the training procedure, and the reported evaluation metrics. Regarding our efficency analysis, we made sure to report numbers for the parameter count *and* a measure for the speed other than FLOPs, as recommended by Dehghani et al. (2021). Since runtime is heavily dependant on training parameters such as batch size, we complement this with asymptotic complexity.

Practical Impact and Future Work Our study has an immediate impact on practitioners who seek to employ robust text categorization models in research projects and in industrial operational environments. Furthermore, we advocate to use an MLP baseline in future text categorization research, for which we provide concrete guidelines in Appendix A. As future work, it would be interesting to analyze multi-label classification tasks and to compare with hierarchical text categorization methods (Peng et al., 2018; Xiao et al., 2019). Another interesting yet challenging setting would be few-shot classification (Brown et al., 2020).

7 Conclusion

We argue that a wide multi-layer perceptron enhanced with today's best practices should be considered as a strong baseline for text classification tasks. In fact, the experiments show that our WideMLP is oftentimes on-par or even better than recently proposed models that synthesize a graph structure from the text.

The source code is available online: https://github.com/lgalke/text-clf-baselines

Ethical Considerations

The focus of this work is text classification. Potential risks that apply to text classification in general also apply to this work. Nonetheless, we present alternatives to commonly used pretrained language models, which suffer from various sources of bias due to the large and poorly manageable data used for pretraining (Bender et al., 2021). In contrast, the presented alternatives render full control over the training data and, thus, contribute to circumvent the biases otherwise introduced during pretraining.

References

Markus Bayer, Marc-André Kaufhold, and Christian Reuter. 2021. A survey on data augmentation for text classification. *ArXiv preprint*, abs/2107.03158.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, pages 610–623, New York, NY, USA. Association for Computing Machinery.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2001. Latent dirichlet allocation. In *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pages 601–608. MIT Press.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

George Cybenko. 1989. Approximation by superpositions of a sigmoidal function. *Math. Control. Signals Syst.*, 2(4):303–314.

Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019*, pages 101–109. ACM.

Mostafa Dehghani, Anurag Arnab, Lucas Beyer, Ashish Vaswani, and Yi Tay. 2021. The efficiency misnomer. *ArXiv preprint*, abs/2110.12894.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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)*, pages 4927–4936, Online. Association for Computational Linguistics.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Jure Leskovec. 2021. GNNAutoScale: Scalable and expressive graph neural networks via historical embeddings. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 3294–3304. PMLR.

Quentin Fournier, Gaétan Marceau Caron, and Daniel Aloise. 2021. A practical survey on faster and lighter transformers. *ArXiv preprint*, abs/2103.14636.

Lukas Galke, Florian Mai, Alan Schelten, Dennis Brunsch, and Ansgar Scherp. 2017. Using titles vs. full-text as source for automated semantic document annotation. In *Proceedings of the Knowledge Capture Conference, K-CAP 2017, Austin, TX, USA, December 4-6, 2017*, pages 20:1–20:4. ACM.

Santiago González-Carvajal and Eduardo C. Garrido-Merchán. 2020. Comparing BERT against traditional machine learning text classification. *ArXiv preprint*, abs/2005.13012.

William L. Hamilton. 2020. *Graph Representation Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691, Beijing, China. Association for Computational Linguistics.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

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*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

Ammar Ismael Kadhim. 2019. Survey on supervised machine learning techniques for automatic text classification. *Artif. Intell. Rev.*, 52(1):273–292.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura E. Barnes, and Donald E. Brown. 2019. Text classification algorithms: A survey. *Inf.*, 10(4):150.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 2267–2273. AAAI Press.

Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S. Yu, and Lifang He. 2020. A survey on text classification: From shallow to deep learning. *ArXiv preprint*, abs/2008.00364.

Hanxiao Liu, Zihang Dai, David R. So, and Quoc V. Le. 2021a. Pay attention to MLPs. *ArXiv preprint*, abs/2105.08050.

Xien Liu, Xinxin You, Xiao Zhang, Ji Wu, and Ping Lv. 2020. Tensor graph convolutional networks for text classification. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8409–8416. AAAI Press.

Yonghao Liu, Renchu Guan, Fausto Giunchiglia, Yanchun Liang, and Xiaoyue Feng. 2021b. Deep attention diffusion graph neural networks for text classification. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8142–8152, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Haonan Lu, Seth H. Huang, Tian Ye, and Xiuyan Guo. 2019. Graph star net for generalized multi-task learning. *ArXiv preprint*, abs/1906.12330.

Shengfei Lyu and Jiaqi Liu. 2020. Combine convolution with recurrent networks for text classification. *ArXiv preprint*, abs/2006.15795.

Florian Mai, Lukas Galke, and Ansgar Scherp. 2018. Using deep learning for title-based semantic subject indexing to reach competitive performance to full-text. In *Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries, JCDL 2018, Fort Worth, TX, USA, June 03-07, 2018*, pages 169–178. ACM.

Luke Melas-Kyriazi. 2021. Do you even need attention? A stack of feed-forward layers does surprisingly well on ImageNet. *ArXiv preprint*, abs/2105.02723.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119.

Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2021. Deep learning-based text classification: A comprehensive review. *ACM Comput. Surv.*, 54(3):62:1–62:40.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. 2020. Deep double descent: Where bigger models and more data hurt. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. 2018. Towards understanding the role of over-parametrization in generalization of neural networks. *ArXiv preprint*, abs/1805.12076.

Yukio Ohsawa, Nels E. Benson, and Masahiko Yachida. 1998. Keygraph: Automatic indexing by co-occurrence graph based on building construction metaphor. In *Proceedings of the IEEE Forum on Research and Technology Advances in Digital Libraries, IEEE ADL '98, Santa Barbara, California, USA, April 22-24, 1998*, pages 12–18. IEEE Computer Society.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, Ann Arbor, Michigan. Association for Computational Linguistics.

Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 1063–1072. ACM.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Rahul Ragesh, Sundararajan Sellamanickam, Arun Iyer, Ramakrishna Bairi, and Vijay Lingam. 2021. HeteGCN: Heterogeneous graph convolutional networks for text classification. In *WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, Virtual Event, Israel, March 8-12, 2021*, pages 860–868. ACM.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. In *Text Mining*, chapter 1, pages 1–20. John Wiley & Sons, Ltd.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv preprint*, abs/1910.01108.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *ArXiv preprint*, abs/1811.05868.

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, Melbourne, Australia. Association for Computational Linguistics.

Koustuv Sinha, Robin Jia, Dieuwke Hupkes, Joelle Pineau, Adina Williams, and Douwe Kiela. 2021. Masked language modeling and the distributional hypothesis: Order word matters pre-training for little. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2888–2913, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170, Online. Association for Computational Linguistics.

Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. PTE: predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pages 1165–1174. ACM.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020. Efficient transformers: A survey. *ArXiv preprint*, abs/2009.06732.

Ilya O. Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. 2021. MLP-Mixer: An all-MLP architecture for vision. *ArXiv preprint*, abs/2105.01601.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz

Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Ishwar Venugopal, Jessica Töllich, Michael Fairbank, and Ansgar Scherp. 2021. A comparison of deep-learning methods for analysing and predicting business processes. In *International Joint Conference on Neural Networks, IJCNN 2021, Shenzhen, China, July 18-22, 2021*, pages 1–8. IEEE.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Ruishuang Wang, Zhao Li, Jian Cao, Tong Chen, and Lei Wang. 2019b. Convolutional recurrent neural networks for text classification. In *International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019*, pages 1–6. IEEE.

Yequan Wang, Aixin Sun, Jialong Han, Ying Liu, and Xiaoyan Zhu. 2018. Sentiment analysis by capsules. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 1165–1174. ACM.

Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6861–6871. PMLR.

Lingfei Wu, Yu Chen, Kai Shen, Xiaojie Guo, Hanning Gao, Shucheng Li, Jian Pei, and Bo Long. 2021. Graph neural networks for natural language processing: A survey. *ArXiv preprint*, abs/2106.06090.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv preprint*, abs/1609.08144.

Huiru Xiao, Xin Liu, and Yangqiu Song. 2019. Efficient path prediction for semi-supervised and weakly supervised hierarchical text classification. In

The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019, pages 3370–3376. ACM.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 7370–7377. AAAI Press.

Rahul Yedida, Xueqi Yang, and Tim Menzies. 2021. When SIMPLE is better than complex: A case study on deep learning for predicting Bugzilla issue close time. *ArXiv preprint*, abs/2101.06319.

Dell Zhang, Jun Wang, Emine Yilmaz, Xiaoling Wang, and Yuxin Zhou. 2016. Bayesian performance comparison of text classifiers. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*, pages 15–24. ACM.

Lu Zhang, Jiandong Ding, Yi Xu, Yingyao Liu, and Shuigeng Zhou. 2021. Weakly-supervised text classification based on keyword graph. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2803–2813, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3485–3495, Osaka, Japan. The COLING 2016 Organizing Committee.

Xujuan Zhou, Raj Gururajan, Yuefeng Li, Revathi Venkataraman, Xiaohui Tao, Ghazal Bargshady, Prabal Datta Barua, and Srinivas Kondalsamy-Chennakesavan. 2020. A survey on text classification and its applications. *Web Intell.*, 18(3):205–216.

A   Practical Guidelines for Designing a WideMLP

On the basis of our results, we provide recommendations for designing a WideMLP baseline.

Tokenization   We recommend using modern subword tokenizers such as BERT-like WordPiece or SentencePiece that yield a high coverage while needing a relatively small vocabulary.

Input Representation   In contrast to conventional wisdom (Iyyer et al., 2015), we find that pretrained embeddings, e. g., GloVe, can have a detrimental effect when compared to using an MLP with one wide hidden layer. Such an MLP circumvents the bottleneck of the small dimensionality of word embeddings and has a higher capacity.

Depth vs. Width   In text classification, width seems more important than depth. We recommend to use a single, wide hidden layer, i. e., one input-to-hidden and one hidden-to-output layer, e. g., with 1,024 hidden units and ReLU activation. While this might be overparameterized for single-label text classification tasks with few classes, we rely on recent findings that overparameterization leads to better generalization (Neyshabur et al., 2018; Nakkiran et al., 2020).

We further motivate the choice of using wide layers with results from multi-label text classification (Galke et al., 2017), which has shown that a (wide) MLP outperforms all tested classical baselines such as SVMs, k-Nearest Neighbors, and logistic regression. Follow-up work (Mai et al., 2018) then found that also CNN and LSTM do not substantially improve over the wide MLP.

Having a fully-connected layer on-top of a bag-of-words leads to a high number of learnable parameters. Still, the wide first input-to-hidden layer can be implemented efficiently by using an embedding layer followed by aggregation, which avoids large matrix multiplications.

In our experiments, we did not observe any improvement with more hidden layers (WideMLP-2), as suggested by Iyyer et al. (2015), but it might be beneficial for other, more challenging datasets.

Optimization and Regularization   We seek to find an optimization strategy that does not require dataset-specific hyperparameter tuning. This comprises optimizing cross-entropy with Adam (Kingma and Ba, 2015) and default learning rate 10^{-3}, a linearly decaying learning rate schedule and training for a high amount of steps (Nakkiran et al., 2020) (we use 100 epochs) with small batch sizes (we use 16) for sufficient stochasticity.

For regularization during this prolonged training, we suggest to use a high dropout ratio of 0.5. Regarding initialization, we rely on framework defaults, i. e., $\mathcal{N}(0,1)$ for the initial embedding layer and random uniform $\mathcal{U}(-\sqrt{d_{\text{input}}}, \sqrt{d_{\text{output}}})$ for subsequent layers' weight and bias parameters.

B   Connection between BoW-MLP and TextGCN

TextGCN uses the PMI matrix to set up edge weights for word-word connections. A single layer Text-GCN is a BoW-MLP, except for the document embedding. The one-hop neighbors are words which are aggregated after a nonlinear transform. The basic GCN equation $\boldsymbol{H} = \sigma(\hat{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{W})$ reveals that the order of transformation and neighborhood aggregation is irrelevant. The document embedding implies that TextGCN is a semisupervised technique. Truly new documents, as in inductive learning scenarios, would need a special treatment such as using an all zero embedding vector.

A two-layer MLP can be characterized by the equation $\hat{y} = \boldsymbol{W}^{(2)}\sigma(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}) + \boldsymbol{b}^{(2)}$. On bag-of-words inputs, the first layer $\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}$ can be replaced by an equivalent embedding layer with weighting (e. g., TF-IDF or length normalization) being applied during aggregation of the embedding vectors.

The first layer of TextGCN is equivalent to aggregating embedding vectors. A standard GCN layer with shared weights has the form (assuming self-loops have been inserted)

$$\boldsymbol{h}_i = \sum_{j \in N(i)} a_{ij}\boldsymbol{W}^{(1)}\boldsymbol{x}_j + \boldsymbol{b}^{(1)}$$

Now in TextGCN node features are given by the identity, such that $\boldsymbol{x}_j$ is one-hot. Then we can rewrite the first layer of Text-GCN as an aggregation of embeddings $\boldsymbol{E}$. We gain

$$\boldsymbol{h}_i = \sum_{j \in N(i)} a_{ij}\boldsymbol{E}_j$$

as $\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ may again be replaced by an embedding matrix if applied to one hot vectors $\boldsymbol{x}$. Now $\boldsymbol{E}$ contains two types of embedding vectors: word embeddings and document embeddings corresponding to word nodes and document nodes. We see that the

first layer of TextGCN is essentially an aggregation of word embeddings plus the document embedding. Only with a second layer, TextGCN considers the embedding of other documents whose words are connected to the present documents' words.

C Equivalence of Micro-F1 and Accuracy in Multiclass Classification

In multiclass classification, we have a single true label for each instance and the predictions are constrained to a single prediction per instance. As a consequence, the measures accuracy and Micro-F1 coincide to the same formula.

Micro-F1 aggregates true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) globally. It can be expressed as:

$$\text{Micro-F}_1 = \frac{2 \sum_c \text{TP}_c}{2 \sum_c \text{TP}_c + \sum_c \text{FP}_c + \sum_c \text{FN}_c},$$

where c iterates over all classes.

While the accuracy can be expressed as:

$$\text{Acc} = \frac{\sum_c \text{TP}_c + \sum_c \text{TN}_c}{\sum_c \text{TP}_c + \sum_c \text{TN}_c + \sum_c \text{FP}_c + \sum_c \text{FN}_c}$$

In multiclass classification, every true positive is also a true negative for all other classes. When summing those up over the entire dataset, we obtain

$$\sum_c \text{TP}_c = \sum_c \text{TN}_c.$$

Thus, we can rewrite

$$2 \sum_c \text{TP}_c = \sum_c \text{TP}_c + \sum_c \text{TN}_c$$

and see that the Micro-F1 and accuracy are equivalent in the multiclass (a.k.a. single-label) case.