"Low-Resource" Text Classification: A Parameter-Free Classification Method with Compressors

Zhiying Jiang[1,2], **Matthew Y.R. Yang**[1], **Mikhail Tsirlin**[1], **Raphael Tang**[1], **Yiqin Dai**[2] and **Jimmy Lin**[1]

[1] University of Waterloo [2] AFAIK

{zhiying.jiang, m259yang, mtsirlin, r33tang}@uwaterloo.ca
quinn@afaik.io jimmylin@uwaterloo.ca

Abstract

Deep neural networks (DNNs) are often used for text classification due to their high accuracy. However, DNNs can be computationally intensive, requiring millions of parameters and large amounts of labeled data, which can make them expensive to use, to optimize, and to transfer to out-of-distribution (OOD) cases in practice. In this paper, we propose a non-parametric alternative to DNNs that's easy, lightweight, and universal in text classification: a combination of a simple compressor like *gzip* with a k-nearest-neighbor classifier. Without any training parameters, our method achieves results that are competitive with non-pretrained deep learning methods on six in-distribution datasets. It even outperforms BERT on all five OOD datasets, including four low-resource languages. Our method also excels in the few-shot setting, where labeled data are too scarce to train DNNs effectively. Code is available at https://github.com/bazingagin/npc_gzip.

1 Introduction

Text classification, as one of the most fundamental tasks in natural language processing (NLP), has improved substantially with the help of neural networks (Li et al., 2022). However, most neural networks are data-hungry, the degree of which increases with the number of parameters. Hyperparameters must be carefully tuned for different datasets, and the preprocessing of text data (e.g., tokenization, stop word removal) needs to be tailored to the specific model and dataset. Despite their ability to capture latent correlations and recognize implicit patterns (LeCun et al., 2015), complex deep neural networks may be overkill for simple tasks such as topic classification, and lighter alternatives are usually good enough. For example, Adhikari et al. (2019b) find that a simple long short-term memory network (LSTM; Hochreiter and Schmidhuber, 1997) with appropriate regularization can achieve competitive results. Shen et al. (2018) further show that even word-embedding-based methods can achieve results comparable to convolutional neural networks (CNNs) and recurrent neural networks (RNNs).

Among all the endeavors for a lighter alternative to DNNs, one stream of work focuses on using compressors for text classification. There have been several studies in this field (Teahan and Harper, 2003; Frank et al., 2000), most of them based on the intuition that the minimum cross entropy between a document and a language model of a class built by a compressor indicates the class of the document. However, previous works fall short of matching the quality of neural networks.

Addressing these shortcomings, we propose a text classification method combining a lossless compressor, a compressor-based distance metric with a k-nearest-neighbor classifier (kNN). It utilizes compressors in capturing regularity, which is then translated into similarity scores by a compressor-based distance metric. With the resulting distance matrix, we use kNN to perform classification. We carry out experiments on seven in-distribution datasets and five out-of-distribution ones. With a simple compressor like *gzip*, our method achieves results competitive with those of DNNs on six out of seven datasets and outperforms all methods including BERT on all OOD datasets. It also surpasses all models by a large margin under few-shot settings.

Our contributions are as follows: (1) we are the first to use NCD with kNN for topic classification, allowing us to carry out comprehensive experiments on large datasets with compressor-based methods; (2) we show that our method achieves results comparable to non-pretrained DNNs on six out of seven in-distribution datasets; (3) on OOD datasets, we show that our method outperforms all methods, including pretrained models such as BERT; and (4) we demonstrate that our method excels in the few-shot setting of scarce labeled data.

Findings of the Association for Computational Linguistics: ACL 2023, pages 6810–6828
July 9-14, 2023

2 Related Work

2.1 Compressor-Based Text Classification

Text classification using compressors can be divided into two main approaches: (1) Using a compressor to estimate entropy based on Shannon Information Theory; (2) Using a compressor to approximate Kolmogorov complexity and *information distance*.[1]

The first approach mainly employs a text compression technique called Prediction by Partial Matching (PPM)[2] for topic classification. This approach estimates the cross entropy between the probability distribution of a specific class c and a given document d: $H_c(d)$ (Frank et al., 2000; Teahan and Harper, 2003). The intuition is that the lower the cross entropy, the more likely that d belongs to c. Marton et al. (2005); Coutinho and Figueiredo (2015); Kasturi and Markov (2022) further improve the final accuracy by improving the representation to better cope with the compressor.

Another line of compressor-based methods (Khmelev and Teahan, 2003; Keogh et al., 2004) takes advantage of the *information distance* (Bennett et al., 1998), a distance metric derived from Kolmogorov complexity. The intuition of *information distance* is that for two similar objects, there exists a *simple* program to convert one to another. However, most previous works focus on clustering (Vitányi et al., 2009), plagiarism detection (Chen et al., 2004) and time series data classification (Keogh et al., 2004). Few (Marton et al., 2005; Coutinho and Figueiredo, 2015) explore its application to topic classification, and none applies the combination of *information distance* and k-nearest-neighbor (kNN) classifier when $k > 1$ to topic classification. Besides, to the best of our knowledge, all the previous works use relatively small datasets like 20News and Reuters-10. There is neither a comparison between compressor-based methods and deep learning methods nor a comprehensive study of large datasets.

2.2 Deep Learning for Text Classification

The deep learning methods used for text classification can be divided into two: transductive learning, represented by Graph Convolutional Networks (GCNs) (Yao et al., 2019), and inductive learning, dominated by recurrent neural networks (RNNs) and convolutional neural networks (CNNs). We focus on inductive learning in this paper as transductive learning assumes the test dataset is presented during the training, which is not a common scenario in practice.

Zhang et al. (2015) use the character-based CNN with millions of parameters for text classification. Conneau et al. (2017) extend the idea with more layers. Along the line of RNNs, Kawakami (2008) introduce a method that uses LSTMs (Hochreiter and Schmidhuber, 1997) to learn the sequential information for classification. To better capture the important information regardless of position, Wang et al. (2016) incorporate the attention mechanism into the relation classification. Yang et al. (2016) include a hierarchical structure for sentence-level attention. As the parameter number and the model complexity increase, Joulin et al. (2017) look for using a simple linear model with a hidden layer coping with n-gram features and hierarchical softmax to improve efficiency.

The landscape of classification has been further transformed by the widespread use of pretrained models like BERT (Kenton and Toutanova, 2019), with hundreds of millions of parameters pretrained on a corpus containing billions of tokens. BERT achieves the state of the art on text classification (Adhikari et al., 2019a). Built on BERT, Reimers and Gurevych (2019) calculate semantic similarity between pairs of sentences efficiently by using a siamese network architecture and fine-tuning on multiple NLI datasets (Bowman et al., 2015; Williams et al., 2018). We compare *gzip* with these deep learning models.

3 Our Approach

Our approach consists of a lossless compressor, a compressor-based distance metric, and a k-Nearest-Neighbor classifier. Lossless compressors aim to represent information using as few bits as possible by assigning shorter codes to symbols with higher probability. The intuition of using compressors for classification is that (1) compressors are good at capturing regularity; (2) objects from the same category share more regularity than those from different categories. For example, x_1 below belongs to the same category as x_2, but a different category from x_3. If we use $C(\cdot)$ to represent com-

[1] This doesn't indicate these two lines of work are completely parallel. In fact, the expected value of Kolmogorov complexity equals Shannon entropy, up to a constant.

[2] PPM is a text compression scheme utilizing language modeling to estimate cross entropy.

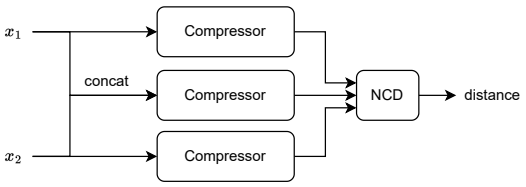

Figure 1: Our approach overview.

pressed length, we will find $C(x_1x_2) - C(x_1) < C(x_1x_3) - C(x_1)$ where $C(x_1x_2)$ means the compressed length of concatenation of x_1 and x_2. In other words, $C(x_1x_2) - C(x_1)$ can be interpreted as how many bytes do we still need to encode x_2 based on the information of x_1:

> x_1 = *Japan's Seiko Epson Corp. has developed a 12-gram flying microrobot.*
>
> x_2 = *The latest tiny flying robot has been unveiled in Japan.*
>
> x_3 = *Michael Phelps won the gold medal in the 400 individual medley.*

This intuition can be formalized as a distance metric derived from Kolmogorov complexity (Kolmogorov, 1963). Kolmogorov complexity $K(x)$ characterizes the length of the shortest binary program that can generate x. $K(x)$ is theoretically the ultimate lower bound for information measurement. To measure information content shared between two objects, Bennett et al. (1998) define *information distance* $E(x, y)$ as the length of the shortest binary program that converts x to y:

$$E(x, y) = \max\{K(x|y), K(y|x)\} \quad (1)$$
$$= K(xy) - \min\{K(x), K(y)\} \quad (2)$$

As the incomputable nature of Kolmogorov complexity renders E(x,y) incomputable, Li et al. (2004) proposes a normalized and computable version of information distance, *Normalized Compression Distance* (NCD), utilizing compressed length $C(x)$ to approximate Kolmogorov complexity $K(x)$. Formally, it's defined as follows (detailed derivation is shown in Appendix A):

$$NCD(x, y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}} \quad (3)$$

The intuition behind using compressed length is that the length of x that has been maximally compressed by a compressor is close to $K(x)$. Generally, the higher the compression ratio, the closer $C(x)$ is to $K(x)$.

```python
import gzip
import numpy as np
for (x1, _) in test_set:
  Cx1 = len(gzip.compress(x1.encode()))
  distance_from_x1 = []
  for (x2, _) in training_set:
    Cx2 = len(gzip.compress(x2.encode())
    x1x2 = " ".join([x1, x2])
    Cx1x2 = len(gzip.compress(x1x2.
    encode())
    ncd = (Cx1x2 - min(Cx1,Cx2)) / max(
    Cx1, Cx2)
    distance_from_x1.append(ncd)
  sorted_idx = np.argsort(np.array(
    distance_from_x1))
  top_k_class = training_set[sorted_idx
    [:k], 1]
  predict_class = max(set(top_k_class),
    key=top_k_class.count)
```

Listing 1: Python Code for Text Classification with *gzip*.

As our main experiment results use *gzip* as the compressor, $C(x)$ here means the length of x after being compressed by *gzip*. $C(xy)$ is the compressed length of concatenation of x and y. With the distance matrix NCD provides, we can then use k-nearest-neighbor to perform classification.

Our method can be implemented with 14 lines of Python code below. The inputs are *training_set*, *test_set*, both consisting of an array of *(text, label)* tuples, and *k* as shown in Listing 1.

Our method is a simple, lightweight, and universal alternative to DNNs. It's simple because it doesn't require any preprocessing or training. It's lightweight in that it classifies without the need for parameters or GPU resources. It's universal as compressors are data-type agnostic, and non-parametric methods do not bring underlying assumptions.

4 Experimental Setup

4.1 Datasets

We choose a variety of datasets to investigate the effects of the number of training samples, the number of classes, the length of the text, and the difference in distribution on accuracy. The details of each dataset are listed in Table 1. Previous works on text classification have two disjoint preferences when choosing evaluation datasets: CNN and RNN-based methods favor large-scale datasets (AG News, DBpedia, YahooAnswers), whereas transductive methods like graph convolutional neural networks focus on smaller ones (20News, Ohsumed, R8, R52) (Li et al., 2022).

Dataset	N_{train}	N_{test}	C	W	L	V
AG News	120K	7.6K	4	44	236	128K
DBpedia	560K	70K	14	54	301	1M
YahooAnswers	1.4M	60K	10	107	520	1.5M
20News	11K	7.5K	20	406	1902	277K
ohsumed	3.4K	4K	23	212	1273	55K
R8	5.5K	2.2K	8	102	587	24K
R52	6.5K	2.6K	52	110	631	26K
KinyarwandaNews	17K	4.3K	14	232	1872	240K
KirundiNews	3.7K	923	14	210	1722	63K
DengueFilipino	4K	500	5	10	62.7	13K
SwahiliNews	22.2K	7.3K	6	327	2.2K	570K
SogouNews	450K	60K	5	589	2.8K	611K

Table 1: Details of datasets used for evaluation. $N_{\{\text{train,test}\}}$ denote the number of training and test set examples, C is the number of classes, W the average number of words in each example, L the average number of characters, and V the vocabulary size.

Model	# Par.	PT	TT	ED	Preprocessing Details
TFIDF+LR	260K	✗	✓	✗	tok+tfidf+dict (+lower)
LSTM	5.2M	✗	✓	✗	tok+dict (+emb+lower+pad)
Bi-LSTM+Attn	8.2M	✗	✓	✗	tok+dict (+emb+lower+pad)
HAN	30M	✗	✓	✗	tok+dict (+emb+lower+pad)
charCNN	2.7M	✗	✓	✗	dict (+lower+pad)
textCNN	31M	✗	✓	✗	tok+dict (+emb+lower+pad)
RCNN	19M	✗	✓	✗	tok+dict (+emb+lower+pad)
VDCNN	14M	✗	✓	✗	dict (+lower+pad)
fastText	8.2M	✗	✓	✗	tok+dict (+lower+pad+$n$gram)
BERT-base	109M	✓	✓	✓	tok+dict+pe (+lower+pad)
W2V	0	✓	✗	✗	tok+dict (+lower)
SentBERT	0	✓	✗	✓	tok+dict (+lower)
TextLength	0	✗	✗	✗	✗
gzip (ours)	0	✗	✗	✗	✗

Table 2: Models with their respective number of *training* parameters, whether they use pre-training (PT), task-specific training (TT)/fine-tuning in BERT, and external data (ED), as well as text preprocessing details.

We include datasets on both sides in order to investigate how our method performs in both situations. Apart from dataset sizes, we also take the number of classes into account by intentionally including datasets like R52 to evaluate the performance of datasets with a large number of classes. We also include the text length of each dataset in Table 1 as previous works (Marton et al., 2005) indicate that it affects the accuracy of compressor-based methods.

Generalizing to out-of-distribution datasets has always been a challenge in machine learning. Even with the success of pretrained models, this problem is not alleviated. In fact, Yu et al. (2021) have shown that improved in-distribution accuracy on pretrained models may lead to poor OOD performance in image classification. In order to compare our method with pretrained models on OOD datasets, we choose five datasets that are unseen in BERT's pretrained corpus—Kinyarwanda news, Kirundi news, Filipino dengue, Swahili news, and Sogou news. Those datasets are chosen to have Latin script which means they have a very similar alphabet as English. For example, Swahili has the same vowels as English but doesn't have `q`, `x` as consonants; Sogou news is in Pinyin – a phonetic romanization of Chinese. Therefore, those datasets can be viewed as permutations of English alphabets (see Table 7 for text examples).

4.2 Baselines

We compare our result with (1) neural network methods that require training and (2) non-parametric methods that use the kNN classifier directly, with or without pre-training on external data. Specifically, we choose mainstream architectures for text classification, like logistic regression, fastText (Joulin et al., 2017), RNNs with or without attention (vanilla LSTM (Hochreiter and Schmidhuber, 1997), bidirectional LSTMs (Schuster and Paliwal, 1997) with attention (Wang et al., 2016), hierarchical attention networks (Yang et al., 2016)), CNNs (character CNNs (Zhang et al., 2015), recurrent CNNs (Lai et al., 2015), very deep CNNs (Conneau et al., 2017)) and BERT (Devlin et al., 2019).

We also include three other non-parametric methods: word2vec (W2V) (Mikolov et al., 2013), pretrained sentence BERT (SentBERT) (Reimers and Gurevych, 2019), and the length of the instance (TextLength), all using a kNN classifier. "TextLength" is a baseline where the text length of the instance is used as the only input into a kNN classifier, whose result rules out the impact of text length in classification.

We present details of models in Table 2. Here we use AG News as an example to estimate the model size, as the number of parameters is affected by the number of classes and the vocabulary size. This dataset has a relatively small vocabulary size and number of classes, making the estimated number of parameters the lower bound of the studied datasets. Some methods require pre-training either on the target dataset or on other external datasets.

We also list preprocessing required by the models in Table 2, including tokenization ("tok"), building vocabulary dictionaries and mapping tokens ("dict"), using pretrained word embeddings ("emb"), lowercasing words ("lower") and padding sequences to a certain length ("pad"). Other model-specific preprocessing includes an extra bag of n-grams ("$n$gram") for *fastText* and positional embedding ("pe") for BERT. Note that for models that only require training, we do not use pretrained word embeddings; otherwise, the boundary between pre-training and training will become ambiguous.

Model	Pre-training	Training	AGNews	DBpedia	YahooAnswers	20News	Ohsumed	R8	R52
TFIDF+LR	✗	✓	0.898	0.982	0.715	0.827	0.549	0.949	0.874
LSTM	✗	✓	0.861	0.985	0.708	0.657	0.411	0.937	0.855
Bi-LSTM+Attn	✗	✓	0.917	0.986	0.732	0.667	0.481	0.943	0.886
HAN	✗	✓	0.896	0.986	0.745	0.646	0.462	0.960	0.914
charCNN	✗	✓	0.914	0.986	0.712	0.401	0.269	0.823	0.724
textCNN	✗	✓	0.817	0.981	0.728	0.751	0.570	0.951	0.895
RCNN	✗	✓	0.912	0.984	0.702	0.716	0.472	0.810	0.773
VDCNN	✗	✓	0.913	0.987	0.734	0.491	0.237	0.858	0.750
fastText	✗	✓	0.911	0.978	0.702	0.690	0.218	0.827	0.571
BERT	✓	✓	0.944	0.992	0.768	0.868	0.741	0.982	0.960
W2V	✓	✗	0.892	0.961	0.689	0.460	0.284	0.930	0.856
SentBERT	✓	✗	0.940	0.937	0.782	0.778	0.719	0.947	0.910
TextLength	✗	✗	0.275	0.093	0.105	0.053	0.090	0.455	0.362
gzip (ours)	✗	✗	0.937	0.970	0.638	0.685	0.521	0.954	0.896

Table 3: Test accuracy compared with *gzip*, red highlighting the ones outperformed by *gzip*. We report results getting from our own implementation. We also include previously reported results for reference in Appendix E.

Dataset	average	*gzip*
AGNews	0.901	**0.937**
DBpedia	0.978	0.970
YahooAnswers	0.726	0.638
20News	0.678	**0.685**
Ohsumed	0.470	**0.521**
R8	0.914	**0.954**
R52	0.838	**0.896**

Table 4: Test accuracy comparison between the average of all baseline models (excluding TextLength) and *gzip*.

5 Results

5.1 Result on in-distribution Datasets

We train all baselines on seven datasets (training details are in Appendix C) using their full training sets. The results are shown in Table 3. Our method performs particularly well on AG News, R8, and R52. On the AG News dataset, fine-tuning BERT yields the highest performance among all methods, while our method, without any pre-training, achieves competitive results, with only 0.007 points lower than BERT. On both R8 and R52, the only non-pretrained neural networks that outperform our method is HAN. For YahooAnswers, the accuracy of *gzip* is about 7% lower than the average neural methods. This may be due to the large vocabulary size of YahooAnswers, which makes it hard for the compressor to compress (detailed discussion is in Appendix F).

Overall, BERT-based models are robust to the size of in-distribution datasets. Character-based models like charCNN and VDCNN perform badly when the dataset is small and the vocabulary size is large (e.g., 20News). Word-based models are better at handling big vocabulary sizes. The result of TextLength is extremely low, indicating the compressed length used in NCD does not benefit from the length distribution of different classes.

gzip does not perform well on extremely large datasets (e.g., YahooAnswers), but is competitive on medium and small datasets. Performance-wise, the only non-pretrained deep learning model that's competitive to *gzip* is HAN, which surpasses *gzip* on four datasets and still achieves relatively high accuracy when it's beaten by *gzip*, unlike textCNN. The difference is that *gzip* doesn't require training.

We list the average of all baseline models' test accuracy (except TextLength for its very low accuracy) in Table 4. We observe that our method is either higher or close to the average on all but the YahooAnswers dataset.

5.2 Result on out-of-distribution Datasets

On five OOD datasets (Kinyarwanda news, Kirundi news, Filipino dengue, Swahili news and Sogou news), we also select DNNs to cover a wide range of parameter numbers. We discard CNN-based methods due to their inferiority when datasets are small, as shown in both Section 5.1 and Zhang et al. (2015). In addition, we also add BERT pretrained on 104 languages (mBERT). We can see in Table 5 that on languages that mBERT has not been pretrained on (Kinyarwanda, Kirundi, or Pinyin), it is worse than BERT. Compared with non-pretrained ones, pretrained models do not hold their advantage on low-resource languages with smaller data sizes, except for Filipino which shares a large vocabulary with English words. On large OOD datasets (i.e., SogouNews), BERT achieves competitive results with other non-pretrained neural networks.

Model/Dataset	KinyarwandaNews		KirundiNews		DengueFilipino		SwahiliNews		SogouNews	
Shot#	Full	5-shot	Full	5-shot	Full	5-shot	Full	5-shot	Full	5-shot
Bi-LSTM+Attn	0.843	0.253±0.061	0.872	0.254±0.053	0.948	0.369±0.053	0.863	0.357±0.049	0.952	0.534±0.042
HAN	0.820	0.137±0.033	0.881	0.190±0.099	0.981	0.362±0.119	0.887	0.264±0.042	0.957	0.425±0.072
fastText	0.869	0.170±0.057	0.883	0.245±0.242	0.870	0.248±0.108	0.874	0.347±0.255	0.930	0.545±0.053
W2V	0.874	0.281±0.236	0.904	0.288±0.189	0.993	0.481±0.158	0.892	0.373±0.341	0.943	0.141±0.005
SentBERT	0.788	0.292±0.062	0.886	0.314±0.060	0.992	0.629±0.143	0.822	0.436±0.081	0.860	0.485±0.043
BERT	0.838	0.240±0.060	0.879	0.386±0.099	0.979	0.409±0.058	0.897	0.396±0.096	0.952	0.221±0.041
mBERT	0.835	0.229±0.066	0.874	0.324±0.071	0.983	0.465±0.048	0.906	0.558±0.169	0.953	0.282±0.060
gzip (ours)	**0.891**	**0.458**±0.065	**0.905**	**0.541**±0.056	**0.998**	**0.652**±0.048	**0.927**	**0.627**±0.072	**0.975**	**0.649**±0.061

Table 5: Test accuracy on OOD datasets with 95% confidence interval over five trials in five-shot setting.

Without any pre-training or fine-tuning, our method outperforms both BERT and mBERT on all five datasets. In fact, our experiments show that our method outperforms both pretrained and non-pretrained deep learning methods on OOD datasets, which back our claim that our method is universal in terms of dataset distributions. To put it simply, our method is designed to handle unseen datasets: the compressor is data-type-agnostic by nature and non-parametric methods do not introduce inductive bias during training.

5.3 Few-Shot Learning

We further compare our method with deep learning methods under the few-shot setting. We carry out experiments on AG News, DBpedia, and SogouNews across both non-pretrained deep neural networks and pretrained ones. We use n-shot labeled examples per class from the training dataset, where $n = \{5, 10, 50, 100\}$. We choose these three datasets, as their scale is large enough to cover 100-shot settings and they vary in text lengths as well as languages. We choose methods whose trainable parameters range from zero parameters like word2vec and sentence BERT to hundreds of millions of parameters like BERT, covering both word-based models (HAN) and an n-gram one (fastText).

We plot the results in Figure 2 (detailed numbers are shown in Appendix D). As shown, *gzip* outperforms non-pretrained models with $5, 10, 50$ settings on all three datasets. When the number of shots is as few as $n = 5$, *gzip* outperforms non-pretrained models by a large margin: *gzip* is 115% better in accuracy than fastText in the AG News 5-shot setting. In the 100-shot setting, *gzip* also outperforms non-pretrained models on AG News and SogouNews but slightly underperforms on DBpedia.

Previous work (Nogueira et al., 2020; Zhang et al., 2021) show that pretrained models are excellent few-shot learners, which is reflected in our consistently high accuracy of BERT and SentBERT on in-distribution datasets like AG News and DBpedia under few-shot settings.[3] It's worth noting, though, that *gzip* outperforms SentBERT for 50 and 100 shots. However, as shown in the SogouNews results, when the dataset is distinctively different from the pretrained datasets, the inductive bias introduced from the pre-training data leads to a low accuracy of BERT and SentBERT with 10, 50 and 100-shot settings, especially with the 5-shot setting. In general, when the shot number increases, the accuracy difference between *gzip* and deep learning methods becomes smaller. W2V is an exception that has a large variance in accuracy. This is due to the vectors being trained for a limited set of words, meaning that numerous tokens in the test set are unseen and hence out-of-vocabulary.

We further investigate the quality of DNNs and our method in the 5-shot setting on five OOD datasets, tabulating results in Table 5. Under 5-shot setting on OOD datasets, our method excels all the deep learning methods by a huge margin: it surpasses the accuracy of BERT by 91%, 40%, 59%, 58% and 194% and surpasses mBERT's accuracy by 100%, 67%, 40%, 12% and 130% on the corresponding five datasets.[4] The reason behind the outperformance of our method is due to compressors' excellent ability to capture regularity, which is prominent when training becomes moot with very few labeled data for DNNs.

6 Analyses

6.1 Using Other Compressors

As the compressor in our method can actually be replaced by any other compressors, we evaluate the

[3] BERT reaches almost perfect accuracy on DBpedia probably because the data is extracted from Wikipedia, which BERT is pretrained on.

[4] mBERT has much higher accuracy than BERT in the few-shot setting on Filipino and Swahili, where mBERT was pretrained on.

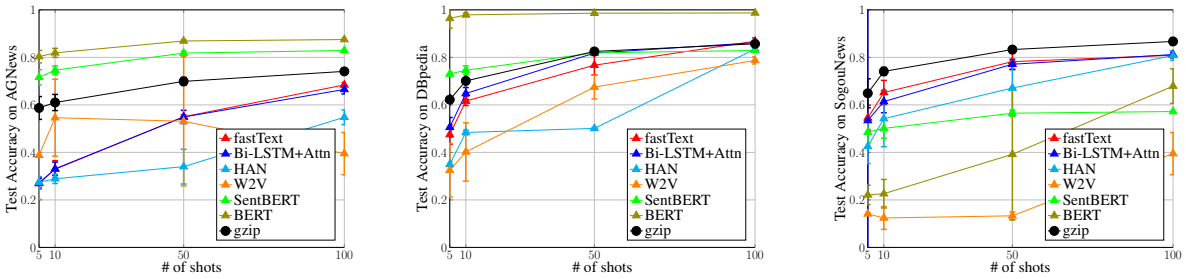

Figure 2: Comparison among different methods using different shots with 95% confidence interval over five trials.

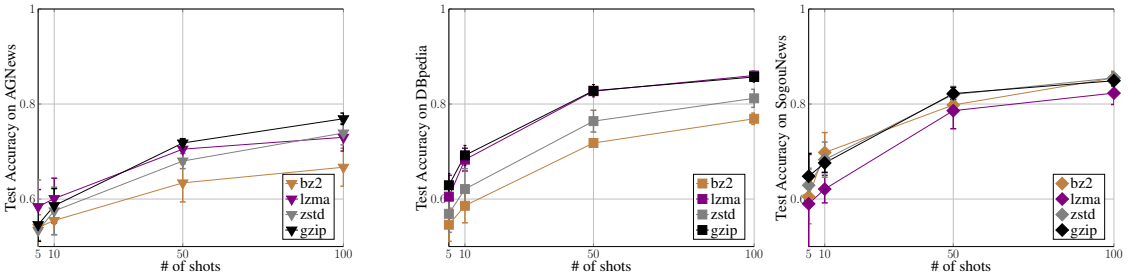

Figure 3: Comparison among different compressors on three datasets with 95% confidence interval over five trials.

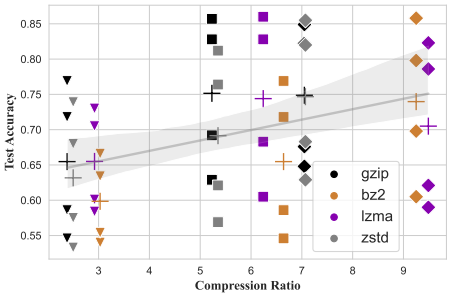

Figure 4: Compression ratio V.S. Test Accuracy across different compressors on three datasets under different shot settings

performance of three other lossless compressors: *bz2*, *lzma*, and *zstandard*. Due to the low compression speed of *lzma*, we randomly select 1,000 test samples from the whole test set to evaluate and conduct our experiments under 5, 10, 50, and 100-shot settings. We repeat the experiments under each setting for five times to calculate the mean and the 95% confidence interval.

Each of the three compressors that we choose has different underlying algorithms from *gzip*. *bz2* uses Burrows-Wheeler algorithm (Burrows, 1994) to permute the order of characters in the strings to create more repeated "substrings" that can be compressed, giving it a higher compression ratio (e.g., it can achieve 2.57 bits-per-character (bpc) on AGNews while *gzip* can achieve only 3.38 bpc). *lzma* is similar to *gzip* in that they are both based on LZ77 (Ziv and Lempel, 1977), a dictionary-based compression algorithm using *(offset, length)* to represent the n-gram that has previously appeared in the search buffer.[5] *zstandard (zstd)* is a new compression algorithm that's built on LZ77, Huffman coding as well as Asymmetric Numeral Systems (ANS) (Duda, 2009). We pick *zstd* because of its high compressing speed and a compression ratio close to *gzip*. A competitive result would suggest that *zstd* might be an alternative to *gzip* and speed up the classification.

In Figure 4, we plot the test accuracy and compression ratio of each compressor. Compression ratio is calculated by $\frac{\text{original size}}{\text{compressed size}}$, so the larger the compression ratio is, the more a compressor can compress.[6] Each marker type represents a dataset, with '+' representing the mean of each compressor's test accuracy across different shot settings.

In general, *gzip* achieves relatively high and stable accuracy across three datasets. *lzma* is competitive with *gzip* but the speed is much slower. Despite its high compression ratio, *bz2* performs the worst across AGNews and DBpedia. Normally, a higher compression ratio of a compressor suggests that the NCD based on it approximates the informa-

[5] *gzip* uses DEFLATE algorithm, which uses Huffman coding (Huffman, 1952) to further encode *(offset, length)* whereas *lzma* uses range coding to do so, resulting *lzma* has a higher compression ratio but a slower compression speed.

[6] We use compression ratio instead of bpc here as the latter one is too close to each other and cannot be differentiated from one another.

Method	AGNews	SogouNews	DBpedia	YahooAnswers
gzip (ce)	0.739±0.046	0.741±0.076	**0.880**±0.010	**0.408**±0.012
gzip (kNN)	**0.752**±0.041	**0.862**±0.033	0.852±0.008	0.352±0.014

Table 6: Comparison with other compressor-based methods under the 100-shot setting.

tion distance $E(x, y)$ better. But in *bz2*'s case, its accuracy is always lower than the regression line (Figure 4). We conjecture it may be because the Burrows-Wheeler algorithm used by *bz2* dismisses the information of character order by permuting characters during compression.

We investigate the correlation between accuracy and compression ratio across compressors and find that they have a moderate monotonic linear correlation as shown in Figure 4. As the shot number increases, the linear correlation becomes more obvious with $r_s = 0.605$ for all shot settings and Pearson correlation $r_p = 0.575, 0.638, 0.691, 0.719$ respectively on 5, 10, 50, and 100-shot settings across four compressors. We have also found that for a single compressor, the easier a dataset can be compressed, the higher the accuracy *gzip* can achieve (details are in Appendix F.1). Combining our findings, we can see that a compressor performs best when it has a high compression ratio on datasets that are highly compressible unless crucial information is disregarded by its compression algorithm.

6.2 Using Other Compressor-Based Methods

A majority of previous compressor-based text classification is built on estimating cross entropy between the probability distribution built on class c and the document d: $H_c(d)$, as we mention in Section 2.1. Summarized in Russell (2010), the procedure of using compressor to estimate $H_c(d)$ is:

1. For each class c, concatenate all samples d_c in the training set belonging to c.
2. Compress d_c as one long document to get the compressed length $C(d_c)$.
3. Concatenate the given test sample d_u with d_c and compress to get $C(d_c d_u)$.
4. The predicted class is $\arg\min_c C(d_c d_u) - C(d_c)$.

The distance metric used by previous work (Marton et al., 2005; Russell, 2010) is mainly $C(d_c d_u) - C(d_c)$. Although using this distance metric is faster than pair-wise distance matrix computation on small datasets, it has several drawbacks: (1) Most compressors have a limited "size", for *gzip* it's the sliding window size that can be used to search back of the repeated string while for *lzma* it's the dictionary size it can keep a record of. This means that even if there are a large number of training samples, the compressor can't take full advantage of those samples; (2) When d_c is large, compressing $d_c d_u$ can be slow, which parallelization can't solve. These two main drawbacks stop this method from being applied to a really large dataset. Thus, we limit the size of the dataset to 1,000 randomly picked test samples and 100-shot from each class in the training set to compare our method with this method.

In Table 6, "*gzip* (ce)" means using the cross entropy $C(d_c d_u) - C(d_c)$ while "*gzip* (kNN)" refers to our method. We carry out each experiment for five times and calculate the mean and 95% confidence interval. Our method outperforms the cross-entropy method on AGNews and SogouNews.

The reason for the large accuracy gap between the two methods on SogouNews is probably because each instance in SogouNews is very long, and the size of each sample can be 11.2K, which, when concatenated, makes d_c larger than 1,000K under 100-shot setting, while *gzip* typically has 32K window size only. When the search space is tremendously smaller than the size of d_c, the compressor fails to take advantage of all the information from the training set, which renders the compression ineffective. The cross-entropy method does perform very well on YahooAnswers. This might be because on a divergent dataset like YahooAnswers, which is created by numerous online users, concatenating all the samples in a class allows the cross-entropy method to take full advantage of all the information from a single class.

We also test the performance of the compressor-based cross-entropy method on *full* AGNews dataset, as it is a relatively smaller one with a shorter single instance. The accuracy is 0.745, not much higher than the 100-shot setting, which further confirms that using $C(d_c d_u) - C(d_c)$ as a distance metric cannot take full advantage of the large datasets. In general, the result suggests that the compressor-based cross-entropy method is not as advantageous as ours on large datasets.

7 Conclusions and Future Work

In this paper, we use *gzip* with a compressor-based distance metric to do text classification.

Our method achieves an accuracy comparable to non-pretrained neural network classifiers on in-distribution datasets and outperforms both pre-trained and non-pretrained models on out-of-distribution datasets. We also find that our method has greater advantages under few-shot settings.

For future works, we will extend this work by generalizing *gzip* to neural compressors on text, as recent studies (Jiang et al., 2022) show that combining neural compressors derived from deep latent variables models with compressor-based distance metrics can even outperform semi-supervised methods for image classification.

Limitations

As the computation complexity of kNN is $O(n^2)$, when the size of a dataset gets really big, speed becomes one of the limitations of our method. Multi-threads and multi-processes can greatly boost the speed. Lempel-Ziv Jaccard Distance (LZJD) (Raff and Nicholas, 2017), a more efficient version of NCD can also be explored to alleviate the inefficiency problem. In addition, as our purpose is to highlight the trade-off between the simplicity of a model and its performance, we focus on the vanilla version of DNNs, which is already complex enough compared with our method, without add-ons like pretrained embeddings (Pennington et al., 2014). This means we do not exhaust all the techniques one can use to improve DNNs, and neither do we exhaust all the text classification methods in the literature. Furthermore, our work only covers traditional compressors. As traditional compressors are only able to capture the orthographic similarity, they may not be sufficient for harder classification tasks like emotional classification. Fortunately, the ability to compress redundant semantic information may be made possible by neural compressors built on latent variable models (Townsend et al., 2018).

Ethics

Being parameter-free, our method doesn't rely on GPU force but CPU resources only. Thus, it does not bring negative environmental impacts revolving around GPU. In terms of overgeneralization, we conduct our experiments on both in-distribution and out-of-distribution datasets, covering six languages. As compressors are data-type agnostic, they are more inclusive to datasets, which allows us to classify low-resource languages like Kinyarwanda, Kirundi, and Swahili and to mitigate the underexposure problem (Hovy and Spruit, 2016). However, as our method has not been fully explored on datasets other than topic classification, it is very possible that our method makes unexpected classification mistakes on tasks like emotion classification. We encourage the usage of this method in the real world to be limited to topic classification and hope that future work can explore more diverse tasks.

Acknowledgement

This research is supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada, and in part by the Global Water Futures program funded by the Canada First Research Excellence Fund (CFREF).

References

Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019a. Docbert: Bert for document classification. *arXiv preprint arXiv:1904.08398*.

Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019b. Rethinking complex neural network architectures for document classification. In *Proceedings of the 2019 Conference of NAACL-HLT, Volume 1 (Long and Short Papers)*, pages 4046–4051.

Charles H Bennett, Péter Gács, Ming Li, Paul MB Vitányi, and Wojciech H Zurek. 1998. Information distance. *IEEE Transactions on information theory*, 44(4):1407–1423.

Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Michael Burrows. 1994. A block-sorting lossless data compression algorithm. *SRC Research Report, 124*.

Xin Chen, Brent Francia, Ming Li, Brian Mckinnon, and Amit Seker. 2004. Shared information and program plagiarism detection. *IEEE Transactions on Information Theory*, 50(7):1545–1551.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1107–1116.

David Pereira Coutinho and Mario AT Figueiredo. 2015. Text classification using compression-based dissimilarity measures. *International Journal of Pattern Recognition and Artificial Intelligence*, 29(05):1553004.

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.

Jarek Duda. 2009. Asymmetric numeral systems. *arXiv preprint arXiv:0902.0271*.

Eibe Frank, Chang Chui, and Ian H Witten. 2000. Text categorization using compression models.

William Hersh, Chris Buckley, TJ Leone, and David Hickam. 1994. Ohsumed: An interactive retrieval evaluation and new large test collection for research. In *SIGIR'94*, pages 192–201. Springer.

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

Dirk Hovy and Shannon L Spruit. 2016. The social impact of natural language processing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 591–598.

David A Huffman. 1952. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101.

Zhiying Jiang, Yiqin Dai, Ji Xin, Ming Li, and Jimmy Lin. 2022. Few-shot non-parametric learning with deep latent variable model. *Advances in Neural Information Processing Systems (NeurIPS)*.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

Armand Joulin, Edouard Grave, and Piotr Bojanowski Tomas Mikolov. 2017. Bag of tricks for efficient text classification. *EACL 2017*, page 427.

Alexandros Kastanos and Tyler Martin. 2021. Graph convolutional network for swahili news classification. *arXiv preprint arXiv:2103.09325*.

Nitya Kasturi and Igor L Markov. 2022. Text ranking and classification using data compression. In *I (Still) Can't Believe It's Not Better! Workshop at NeurIPS 2021*, pages 48–53. PMLR.

Kazuya Kawakami. 2008. Supervised sequence labelling with recurrent neural networks. *Ph. D. thesis*.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Eamonn Keogh, Stefano Lonardi, and Chotirat Ann Ratanamahatana. 2004. Towards parameter-free data mining. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 206–215.

Dmitry V Khmelev and William J Teahan. 2003. A repetition based measure for verification of text collections and for text categorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 104–110.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Andrei N Kolmogorov. 1963. On tables of random numbers. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 369–376.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature*, 521(7553):436–444.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195.

Ming Li, Xin Chen, Xin Li, Bin Ma, and Paul MB Vitányi. 2004. The similarity metric. *IEEE transactions on Information Theory*, 50(12):3250–3264.

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 Transactions on Intelligent Systems and Technology (TIST)*, 13(2):1–41.

Xien Liu, Song Wang, Xiao Zhang, Xinxin You, Ji Wu, and Dejing Dou. 2020. Label-guided learning for text classification. *arXiv preprint arXiv:2002.10772*.

Evan Dennison Livelo and Charibeth Cheng. 2018. Intelligent dengue infoveillance using gated recurrent neural learning and cross-label frequencies. In *2018 IEEE International Conference on Agents (ICA)*, pages 2–7. IEEE.

Yuval Marton, Ning Wu, and Lisa Hellerstein. 2005. On compression-based text classification. In *European Conference on Information Retrieval*, pages 300–314. Springer.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Rubungo Andre Niyongabo, Qu Hong, Julia Kreutzer, and Li Huang. 2020. Kinnews and kirnews: Benchmarking cross-lingual text classification for kinyarwanda and kirundi. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5507–5521.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pre-trained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718.

Antoine Nzeyimana and Andre Niyongabo Rubungo. 2022. Kinyabert: a morphology-aware kinyarwanda language model. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5347–5363.

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.

Edward Raff and Charles Nicholas. 2017. An alternative to ncd for large sequences, lempel-ziv jaccard distance. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1007–1015.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Stuart J Russell. 2010. *Artificial intelligence a modern approach*. Pearson Education, Inc.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

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.

Eyal Shnarch, Ariel Gera, Alon Halfon, Lena Dankin, Leshem Choshen, Ranit Aharonov, and Noam Slonim. 2022. Cluster & tune: Boost cold start performance in text classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7639–7653.

William J Teahan and David J Harper. 2003. Using compression-based language models for text categorization. In *Language modeling for information retrieval*, pages 141–165. Springer.

James Townsend, Thomas Bird, and David Barber. 2018. Practical lossless compression with latent variables using bits back coding. In *International Conference on Learning Representations*.

Paul MB Vitányi, Frank J Balbach, Rudi L Cilibrasi, and Ming Li. 2009. Normalized information distance. In *Information theory and statistical learning*, pages 45–82. Springer.

Canhui Wang, Min Zhang, Shaoping Ma, and Liyun Ru. 2008. Automatic online news issue construction in web environment. In *Proceedings of the 17th international conference on World Wide Web*, pages 457–466.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the NAACL-HLT, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 7370–7377.

Yaodong Yu, Heinrich Jiang, Dara Bahri, Hossein Mobahi, Seungyeon Kim, Ankit Singh Rawat, Andreas Veit, and Yi Ma. 2021. An empirical study of pre-trained vision models on out-of-distribution generalization. In *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*.

Haode Zhang, Yuwei Zhang, Li-Ming Zhan, Jiaxin Chen, Guangyuan Shi, Xiao-Ming Wu, and Albert YS Lam. 2021. Effectiveness of pre-training

for few-shot intent classification. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1114–1120.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Jacob Ziv and Abraham Lempel. 1977. A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3):337–343.

A Derivation of NCD

Recall that *information distance* $E(x, y)$ is:

$$E(x, y) = \max\{K(x|y), K(y|x)\} \quad (4)$$
$$= K(xy) - \min\{K(x), K(y)\} \quad (5)$$

$E(x, y)$ equates the similarity between two objects in a program that can convert one to another. The simpler the converting program is, the more similar the objects are. For example, the negative of an image is very similar to the original one as the transformation can be simply described as "inverting the color of the image".

In order to compare the similarity, the relative distance is preferred. Vitányi et al. (2009) propose a normalized version of $E(x, y)$ called *Normalized information distance* (NID).

Definition 1 (NID) *NID is a function:* $\Omega \times \Omega \to [0, 1]$*, where* Ω *is a non-empty set, defined as:*

$$NID(x, y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}}. \quad (6)$$

Equation (6) can be interpreted as follows: Given two sequences x, y, $K(y) \geq K(x)$:

$$\text{NID}(x, y) = \frac{K(y) - I(x : y)}{K(y)} = 1 - \frac{I(x : y)}{K(y)}, \quad (7)$$

where $I(x : y) = K(y) - K(y|x)$ means the *mutual algorithmic information*. $\frac{I(x:y)}{K(y)}$ means the shared information (in bits) per bit of information contained in the most informative sequence, and Equation (7) here is a specific case of Equation (6).

Normalized Compression Distance (NCD) is a computable version of NID based on real-world compressors. In this context, $K(x)$ can be viewed as the length of x after being maximally compressed. Suppose we have $C(x)$ as the length of compressed x produced by a real-world compressor, then NCD is defined as:

$$\text{NCD}(x, y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}. \quad (8)$$

NCD is thus computable in that it not only uses compressed length to approximate $K(x)$ but also replaces conditional Kolmogorov complexity with $C(xy)$ that only needs a simple concatenation of x, y.

B Dataset Details

In addition to statistics of the datasets we use, we also include one example for each dataset in Table 7. We then briefly introduce what the dataset is about and how are they collected.

AG News[7] contains more than 1 million news articles from an academic news search engine CometoMyHead and is collected for a research purpose;

DBpedia (Lehmann et al., 2015) is extracted from Wikipedia as a crowd-sourced project and we use the version in torchtext version 0.11.

YahooAnswers is introduced in Zhang et al. (2015) through the Yahoo! Webscope program and use the 10 largest main categories for topic classification corpus.

20News (Lang, 1995) is originally collected by Ken Lang and is widely used to evaluate text classification and we use the version in scikit-learn.

Ohsumed (Hersh et al., 1994) is collected from 270 medical journals over a five-year period (1987-1991) with 23 cardiovascular diseases. We use the subset introduced in (Yao et al., 2019) to create a single-label classification.

Both R8 and R52 are two subsets from Reuters-21578 collection (Joachims, 1998) which can be downloaded from Text Categorization Corpora.

KirundiNews (KirNews) and KinyarwandaNews (KinNews) are introduced in (Niyongabo et al., 2020), collected as a benchmark for text classification on two low-resource African languages, which can be freely downloaded from the repository.

SwahiliNews (Swahili)[8] is a news dataset in Swahili. It's spoken by 100-150 million people across East Africa, and the dataset is created to help leverage NLP techniques across the African continent, which can be freely downloaded from huggingface datasets.

DengueFilipino (Filipino) (Livelo and Cheng, 2018) is a multi-label low-resource classification dataset, which can be freely downloaded from huggingface datasets. We process it as a single-label classification task — we randomly select a label if an instance have multiple labels and use the same processed dataset for every model.

SogouNews is collected by Wang et al. (2008), segmented and labeled by Zhang et al. (2015). We use the version that's publicly available on torchtext.

[7] http://groups.di.unipi.it/gulli/AG_corpus_of_news_articles.html

[8] https://doi.org/10.5281/zenodo.5514203

Dataset	Sample Text
AGNews	“Wall St. Bears Claw Back Into the Black (Reuters) Reuters - Short-sellers, Wall Street’s dwindling band of ultra-cynics, are seeing green again.”
DBpedia	“European Association for the Study of the Liver”, “The European Association for the Study of the Liver (EASL) is a European professional association for liver disease.”
YahooAnswers	“Is a transponder required to fly in class C airspace?”,“I’ve heard that it may not be for some aircraft. What are the rules?”,“the answer is that you must have a transponder in order to fly in a class C airspace.”
20News	“Subject: WHAT car is this!? Nntp-Posting-Host: rac3.wam.umd.edu Organization: University of Maryland, College Park Lines: 15 I was wondering if anyone out there could enlighten me on this car I saw the other day. It was a 2-door sports car, looked to be from the late 60s/ early 70s. It was called a Bricklin. The doors were really small. In addition, the front bumper was separate from the rest of the body. This is all I know. If anyone can tellme a model name, engine specs, years of production, where this car is made, history, or whatever info you have on this funky looking car, please e-mail. Thanks,- IL —- brought to you by your neighborhood Lerxst —-”
Ohsumed	“Protection against allergen-induced asthma by salmeterol.The effects of the long-acting beta 2-agonist salmeterol on early and late phase airways events provoked by inhaled allergen were assessed in a group of atopic asthmatic patients.In a placebo-controlled study, salmeterol 50 micrograms inhaled before allergen challenge ablated both the early and late phase of allergen-induced bronchoconstriction over a 34 h time period.Salmeterol also completely inhibited the allergen-induced rise in non-specific bronchial responsiveness over the same time period.These effects were shown to be unrelated to prolonged bronchodilatation or functional antagonism.These data suggest novel actions for topically active long-acting beta 2-agonists in asthma that extend beyond their protective action on airways smooth muscle.”
R8	“champion products ch approves stock split champion products inc said its board of directors approved a two for one stock split of its common shares for shareholders of record as of april the company also said its board voted to recommend to shareholders at the annual meeting april an increase in the authorized capital stock from five mln to mln shares reuter ”
R52	“january housing sales drop realty group says sales of previously owned homes dropped pct in january to a seasonally adjusted annual rate of mln units the national association of realtors nar said but the december rate of mln units had been the highest since the record mln unit sales rate set in november the group said the drop in january is not surprising considering that a significant portion of december s near record pace was made up of sellers seeking to get favorable capital gains treatment under the old tax laws said the nar s john tuccillo reuter”
KinNews	“mutzig beer fest itegerejwe n’abantu benshi kigali mutzig beer fest thedition izabera juru parki rebero hateganyijwe imodoka zizajya zifata abantu buri minota zibakura sonatubei remera stade kumarembo areba miginai remera mugiporoso hamwe mumujyi rond point nini kigali iki gitaramo kizaba cyatumi-wemo abahanzi batandukanye harimo kizigenza mugihugu cy’u burundi uzwi izina kidum benshi bakaba bamuziho gucuranga neza live music iki gitaramo kikazatangira isaha saa kumi n’ebyiri z’umugoroba taliki kugeza saa munani mugitondo taliki kwinjira bizasaba amafaranga y’u rwanda kubafite mutzig golden card aha niho tike zigurirwa nakumat la gallette simba super market flurep”
KirNews	“sentare yiyungurizo ntahangwa yagumije munyororo abamenyeshamakuru bane abo bamenyeshamakuru bakaba bakorera ikinyamakuru iwacu bakaba batawe mvuto kwezi kw’icumi umwaka bakaba bagiye ntara bubanza kurondera amakuru yavuga hari abagwanya leta binjiye gihugu abajejwe umutekano baciye babafata bagishika komine bukinanyana ahavugwa bagwanyi bakaba baciye bashikirizwa sentare nkuru bubanza umushikirizamanza akaba yaciye abagiriza icaha co kwifatanya n’abagwanyi gutera igihugu icaha cahavuye gihindurwa citwa icaha co gushaka guhungabanya umutekano w’igihugu iyo sentare yaciye ibacira imyaka ibiri nusu n’amande y’amafaranga umuriyoni umwe umwe icabafashe cane n’ubutumwe bwafatanwe umwe muribo buvuga ’bagiye i bubanza gufasha abagwanyi” ababuranira bakaba barageragejeje kwerekana kwabo bamenyeshamakuru ataco bapfana n’abagwanyi ikinyamakuru iwacu kikaba carunguruje sentare yiyungurizo ntahangwa ariko sentare yafashe ingingo kubagumiza mumunyororo ikinyamakuru iwacu kikavuga kigiye kwitura sentare ntahinyuzwa”
Filipino	“Kung hindi lang absent yung ibang pipirma sa thesis namen edi sana tapos na hardbound”
SwahiliNews	“TIMU ya taifa ya Tanzania, Serengeti Boys jana ilijiweka katika nafasi fi nyu katika mashindano ya Mataifa ya Afrika kwa wachezaji wenye umri chini ya miaka 17 baada ya kuchapwa mabao 3-0 na Uganda kwenye Uwanja wa Taifa, Dar es Salaam.Uganda waliandika bao lao la kwanza katika dakika ya 15 lililofungwa na Kawooya Andrew akiunganisha wavuni krosi ya Najibu Viga huku lile la pili likifungwa na Asaba Ivan katika dakika ya 27 Najib Yiga.Serengeti Boys iliendelea kulala, Yiga aliifungia Uganda bao la tatu na la ushindi na kuifanya Serengeti kushika mkia katika Kundi A na kuacha simanzi kwa wapenzi wa soka nchini. Serengeti Boys inasubiri mchezo wa mwisho dhidi ya Senegal huku Nigeria ikisonga mbele baada ya kushinda mchezo wake wa awali kwenye uwanja huo na kufikisha pointi sita baada ya kushinda ule wa ufunguzi dhidi ya Tanzania.”
SogouNews	“2008 di4 qi1 jie4 qi1ng da3o guo2 ji4 che1 zha3n me3i nv3 mo2 te4 ”,“2008di4 qi1 jie4 qi1ng da3o guo2 ji4 che1 zha3n yu2 15 ri4 za4i qi1ng da3o guo2 ji4 hui4 zha3n zho1ng xi1n she4ng da4 ka1i mu4 . be3n ci4 che1 zha3n jia1ng chi2 xu4 da4o be3n yue4 19 ri4 . ji1n nia2n qi1ng da3o guo2 ji4 che1 zha3n shi4 li4 nia2n da3o che2ng che1 zha3n gui1 mo2 zui4 da4 di2 yi1 ci4 , shi3 yo4ng lia3o qi1ng da3o guo2 ji4 hui4 zha3n zho1ng xi1n di2 qua2n bu4 shi4 ne4i wa4i zha3n gua3n . yi3 xia4 we2i xia4n cha3ng mo2 te4 tu2 pia4n .”

Table 7: Sample text for each dataset.

Paper	Model	Emb	AGNews	DBpedia	YahooAnswers	20News	Ohsumed	R8	R52	SogouNews
Zhang et al. (2015)	LSTM	✓	0.860	0.985	0.708	-	-	-	-	0.951
	charCNN	✗	0.914	0.985	0.680	-	-	-	-	0.956
Yang et al. (2016)	HAN	✓	-	-	0.758	-	-	-	-	-
Joulin et al. (2017)	charCNN	✗	0.872	0.983	0.712	-	-	-	-	0.951
	VDCNN	✗	0.913	0.987	0.734	-	-	-	-	0.968
	fastText	✗	0.915	0.981	0.720	-	-	-	-	0.939
Conneau et al. (2017)	VDCNN	✗	0.908	0.986	0.724	-	-	-	-	0.962
Yao et al. (2019)	LSTM	✗	-	-	-	0.657	0.411	0.937	0.855	-
	fastText	✓	-	-	-	0.797	0.557	0.947	0.909	-
Liu et al. (2020)	fastText	✓	0.925	0.986	0.723	0.114	0.146	0.860	0.716	-
	BiLSTM	✓	-	-	-	0.732	0.493	0.963	0.905	-
	BERT	✗	-	-	-	0.679	0.512	0.960	0.897	-

Table 8: Results reported in previous works on datasets with abundant resources with embedding (Emb) information.

Paper	Model	Emb	PT	KinyarwandaNews	KirundiNews	SwahiliNews	DengueFilipino
Niyongabo et al. (2020)	charCNN	✗	✗	0.717	0.692	-	-
	BiGRU	✓(Kin. W2V)	✗	0.887	0.859	-	-
	CNN	✓(Kin. W2V)	✗	0.875	0.857	-	-
Kastanos and Martin (2021)	fastText	✗	✗	-	-	0.675	-
Nzeyimana and Rubungo (2022)	BERT_{BPE}	✗	✓(Kin. Corpus)	0.883	-	-	-
	BERT_{MORPHO}	✗	✓(Kin. Corpus)	0.869	-	-	-
	KinyaBERT	✗	✓(Kin. Corpus)	0.880	-	-	-

Table 9: Results reported in previous works on low resource languages with embedding (Emb) and pre-training (PT) information.

Paper	Model	AGNews	DBpedia
Shnarch et al. (2022)	BERT	0.619	0.312
	$\text{BERT}_{\text{IT:CLUSTER}}$	0.807	0.670

Table 10: Results reported in previous works on 64-sample learning, corresponding to 14-shot for AGNews and ≈5-shot for DBpedia.

C Implementation Details

We use different hyper-parameters for full-dataset settings and few-shot settings.

For both LSTM, Bi-LSTM+Attn, fastText, we use embedding size $= 256$, dropout rate $= 0.3$. For full-dataset setting, the learning rate is set to be 0.001 and decay rate $= 0.9$ for Adam optimizer (Kingma and Ba, 2015), number of epochs $= 20$, with batch size $= 64$; for few-shot setting, the learning rate $= 0.01$, the decay rate $= 0.99$, batch size $= 1$, number of epochs $= 50$ for 50-shot and 100-shot, epoch $= 80$ for 5-shot and 10-shot. For LSTM and Bi-LSTM+Attn, we set RNN layer $= 1$, hidden size $= 64$. For fastText, we use 1 hidden layer whose dimension is set to 10.

For HAN, we use 1 layer for both word-level RNN and sentence-level RNN, the hidden size of both of them are set to 50, and the hidden sizes of both attention layers are set to 100. It's trained with batch size $= 256$, 0.5 decay rate for 6 epochs.

For BERT, the learning rate is set to be $2e{-}5$ and the batch size is set to be 128 for English and SogouNews while for low-resource languages, we set the learning rate to be $1e{-}5$ with batch size to be 16 for 5 epochs. We use publicly available transformers library (Wolf et al., 2020) for BERT and specifically we use `bert-base-uncased` checkpoint for BERT and `bert-base-multilingual-uncased` for mBERT.

For charCNN and textCNN, we use the same hyper-parameters setting in Adhikari et al. (2019b) except when in the few-shot learning setting, we reduce the batch size to 1, reducing the learning rate to $1e-4$ and increase the number of epochs to 60. We also use their open source hedwig repo for implementation. For VDCNN, we use the shallowest 9-layer version with embedding size set to be 16, batch size set to be 64 learning rate set to be $1e-4$ for full-dataset setting, and batch size $= 1$, epoch number $= 60$ for few-shot setting. For RCNN, we use embedding size $= 256$, hidden size of RNN $= 256$, learning rate $= 1e-3$, and the same batch size and epoch setting as VDCNN for full-dataset and few-shot settings.

In general, we perform grid search for hyper-parameters on all the neural network models and we use a test set to validate, which only overestimates the accuracy.

For preprocessing, we don't use any pretrained word embedding for any word-based models. The

reason is that we have a strict categorization between "training" and "pre-training", involving pretrained embedding will make DNNs' categories ambiguous. Neither do we use data augmentation during the training. The procedures of tokenization for both word-level and character-level, padding for batch processing are, however, inevitable.

For all non-parametric methods, the only hyper-parameter is k. We set $k = 2$ for all the methods on all the datasets and we report the maximum possible accuracy getting from the experiments for each method. For Sentence-BERT, we use the `paraphrase-MiniLM-L6-v2` checkpoint.

Our method only requires CPUs and we use 8-core CPUs to take advantage of multi-processing. The time of calculating distance matrix using *gzip* takes about half an hour on AGNews, two days on DBpedia and SogouNews, and six days on YahooAnswers.

D Few-Shot Results

The exact numerical values of accuracy shown in Figure 2 is listed in three tables below.

Dataset	AGNews			
#Shot	5	10	50	100
fastText	0.273±0.021	0.329±0.036	0.550±0.008	0.684±0.010
Bi-LSTM+Attn	0.269±0.022	0.331±0.028	0.549±0.028	0.665±0.019
HAN	0.274±0.024	0.289±0.020	0.340±0.073	0.548±0.031
W2V	0.388±0.186	0.546±0.162	0.531±0.272	0.395±0.089
BERT	0.803±0.026	0.819±0.019	0.869±0.005	0.875±0.005
SentBERT	0.716±0.032	0.746±0.018	0.818±0.008	0.829±0.004
gzip (ours)	0.587±0.048	0.610±0.034	0.699±0.017	0.741±0.007

Table 11: Few-Shot result on AG News

Dataset	DBpedia			
#Shot	5	10	50	100
fastText	0.475±0.041	0.616±0.019	0.767±0.041	0.868±0.014
Bi-LSTM+Attn	0.506±0.041	0.648±0.025	0.818±0.008	0.862±0.005
HAN	0.350±0.012	0.484±0.010	0.501±0.003	0.835±0.005
W2V	0.325±0.113	0.402±0.123	0.675±0.05	0.787±0.015
BERT	0.964±0.041	0.979±0.007	0.986±0.002	0.987±0.001
SentBERT	0.730±0.008	0.746±0.018	0.819±0.008	0.829±0.004
gzip (ours)	0.622±0.022	0.701±0.021	0.825±0.003	0.857±0.004

Table 12: Few-Shot result on DBpedia

Dataset	SogouNews			
#Shot	5	10	50	100
fastText	0.545±0.053	0.652±0.051	0.782±0.034	0.809±0.012
Bi-LSTM+Attn	0.534±0.042	0.614±0.047	0.771±0.021	0.812±0.008
HAN	0.425±0.072	0.542±0.118	0.671±0.102	0.808±0.020
W2V	0.141±0.005	0.124±0.048	0.133±0.016	0.395±0.089
BERT	0.221±0.041	0.226±0.060	0.392±0.276	0.679±0.073
SentBERT	0.485±0.043	0.501±0.041	0.565±0.013	0.572±0.003
gzip (ours)	0.649±0.061	0.741±0.017	0.833±0.007	0.867±0.016

Table 13: Few-Shot result on SogouNews

E Other Reported Results

In Table 3 and Table 5, we report the result from our hyper-parameter setting and implementation. However, we find that we couldn't replicate previously reported results in some cases — we get higher or lower results than previously reported ones, which may be due to different experiment settings (e.g., they may use pretrained word embeddings while we don't) or different hyper-parameter settings. Thus, we provide results reported by some previous papers for reference in Table 8, Table 9 and Table 10. Note that SogouNews is listed in the first table as it has abundant resources and is commonly used as a benchmark for DNNs that excel at large datasets. As the studies carried out in low-resource languages and few-shot learning scenarios are insufficient, in Table 9 and in Table 10, we also report the result of variants of our models like BiGRU using Kinyarwanda embeddings (Kin. W2V) and BERT$_{MORPHO}$ incorporating morphology and pretrained on Kinyarwanda corpus (Kin. Corpus) in addition to models we use in the paper. We don't find any result reported for DengueFilipino as previous works' evaluation uses multi-label metrics.

F Performance Analysis

To understand the merits and shortcomings of using *gzip* for classification, we evaluate *gzip*'s performance in terms of both the absolute accuracy and the relative performance compared to the neural methods. An absolute low accuracy with a high relative performance suggests that the dataset itself is difficult, while a high accuracy with a low relative performance means the dataset is better solved by a neural network. As our method performs well on OOD datasets, we are more interested in analyzing ID cases. We carry out seven in-distribution datasets and one out-of-distribution dataset across fourteen models to account for different ranks. We analyze both the relative performance and the absolute accuracy regarding the vocabulary size and the compression rate of both datasets (i.e., how easily a dataset can be compressed) and compressors (i.e., how well a compressor can compress).

To represent the relative performance with regard to other methods, we use the normalized rank percentage, computed as $\frac{\text{rank of gzip}}{\text{total\#methods}}$; the lower the score, the better *gzip* is. We use "bits per character"(bpc) to evaluate the compression rate. The procedure is to randomly sample a thousand in-

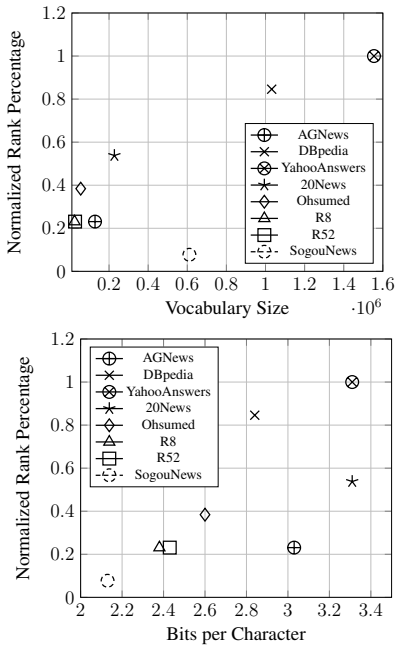

Figure 5: Relative performance v.s. vocabulary size and compression rate.

stances from the training and test set respectively, calculate the compressed length, and divide by the number of characters. Sampling is to keep the size of the dataset constant.

F.1 Relative Performance

Combining Table 1 and Table 3, we see that accuracy is largely unaffected by the average length of a single sample: with the Spearman coefficient $r_s = -0.220$. But the relative performance is more correlated with vocabulary size ($r_s = 0.561$) as we can see in Figure 5. SogouNews is an outlier in the first plot: on a fairly large vocabulary-sized dataset, *gzip* ranks first. The second plot may provide an explanation for that — the compression ratio for SogouNews is high which means even with a relatively large vocabulary size, there is also repetitive information that can be squeezed out. With $r_s = 0.785$ on the correlation between the normalized rank percentage and the compression rate, we can see when a dataset is easier to compress, our method may be a strong candidate as a classifier.

F.2 Absolute Accuracy

Similarly, we evaluate the accuracy of classification with respect to the vocabulary size and we've found there is almost no monotonic relation ($r_s = 0.071$). With regard to bpc, the monotonic relation is not as strong as the one with the rank percentage ($r_s = -0.56$). Considering the effect that vocabulary size has on the relative performance, our method with *gzip* may be more susceptible to the vocabulary size than neural network methods. To distinguish between a "hard" dataset and an "easy" one, we average all models' accuracies. The datasets that has the lowest accuracies are 20News and Ohsumed, which are two datasets that have the longest average length of texts.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 7.

☑ A2. Did you discuss any potential risks of your work?
Section 8.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Section 1.

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 3.

☑ B1. Did you cite the creators of artifacts you used?
Appendix B and C.

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Appendix B and C.

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Appendix B and C.

☑ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Appendix B.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 4.1 and Appendix B.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
In Section 4.1 Table 1.

C ☑ Did you run computational experiments?

Section 4.

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4 and Appendix C.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Appendix C.

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 4.3, 4.4, 4.5.

☐ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Not applicable. Left blank.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.