Speeding Up Transformer Training By Using Dataset Subsampling - An Exploratory Analysis

Lovre Torbarina[†], **Velimir Mihelcic, Bruno Sarlija, Lukasz Roguski, Zeljko Kraljevic**[†]
doXray B.V., Neede, Netherlands
{lovre.torbarina, velimir.mihelcic, bruno.sarlija,
lukasz.roguski, zeljko.kraljevic}@doxray.com

Abstract

Transformer-based models have greatly advanced the progress in the field of the natural language processing and while they achieve state-of-the-art results on a wide range of tasks, they are cumbersome in parameter size. Subsequently, even when pre-trained transformer models are used for fine-tuning on a given task, if the dataset is large, it may still not be feasible to fine-tune the model within a reasonable time. For this reason, we empirically test 8 subsampling methods for reducing the dataset size on text classification task and report the trade-off between metric score and training time. 7 out of 8 methods are simple methods, while the last one is CRAIG, a method for coreset construction for data-efficient model training. We obtain the best result with the CRAIG method, offering an average decrease of 0.03 points in f-score on test set while speeding up the training time on average by 63.93%, relative to the score and time obtained by using the full dataset. Lastly, we show the trade-off between speed and performance for all sampling methods on three different datasets.

1 Introduction

In recent years, a great progress has been observed in the field of natural language processing thanks to use of deep neural network models and, most notably, Transformer (Vaswani et al., 2017) architecture-based models, such as BERT (Devlin et al., 2019). While transformer-based models achieve state-of-the-art results on a wide range of benchmarks (Wang et al., 2018, 2019), they are cumbersome in the number of parameters. Consequently, computers with large amounts of RAM memory and specialized hardware, such as multiple GPU or TPU devices, are required to train the models in reasonable time. As a result, such vast computational requirements put constraints on what can be done in smaller research teams, both in academia and in industry.

As one of the approaches to alleviate this problem, an extensive research has been conducted on methods for model compression, such as, model pruning, quantization, knowledge distillation, parameter sharing and others (Gupta and Agrawal, 2021). Although model compression methods can produce models characterising with low-latency and low-memory footprint for transfer learning, e.g. DistilBERT (Sanh et al., 2020), but if the dataset is significantly large, it may still be not feasible to fine-tune the model within a reasonable time. Therefore, if the model's size can't be further reduced without sacrificing its generalization performance, reducing the dataset is a task worth considering to further reduce training resource requirements.

The challenge of reducing dataset size while trying to preserving model's performance is the main focus of this paper. Previous research conducted on this topic mainly focuses on computer vision or traditional machine learning domain. Pruning the dataset by removing noisy examples to improve model's performance was done in Angelova et al. (2005), while Lapedriza et al. (2013) ranks examples by how *valuable* they are and takes the top ones in a greedy fashion w.r.t. the rankings. In instance selection methods, only the informative examples are selected prior to model training (Olvera-López et al., 2010). Similarly, a coreset (or weighted subset) construction was done in Tsang et al. (2005); Bachem et al. (2017) for specific traditional ML models while Mirzasoleiman et al. (2020) introduced a general approach applicable also to neural network models. In Wang et al. (2020); Sucholutsky and Schonlau (2020), the idea of knowledge distillation was used to distill a dataset to a small set of synthetic examples. To the best of our knowledge, there is no previous work on reducing the train set size for lowering the time

[†]Corresponding authors.

and resource requirements for transformer-based model training.

In this paper, we applied a general method for coreset construction described in Mirzasoleiman et al. (2020) to a transformer model. We empirically compared its performance against simpler subsampling approaches across 3 text classification datasets. The rest of the paper is organized as follows. In section 2, we describe methodology, including construction of the datasets, description of the sampling methods and experiment set up. In section 3, we show the performance across the datasets. And finally, in section 4, we discuss the results together with our outlook for future work.

2 Methods

2.1 Datasets

In our experiments we used 3 datasets for text classification where texts are sorted over time and partitioned (or segmented) by e.g. months, based on their corresponding timestamp. We chose this data ordering because it simulates a typical real-world problem in which new data arrives over time. An overview of the dataset properties is presented in Table 1. Texts that don't have a timestamp have been discarded from each dataset. All of the datasets have a non-uniform distribution of examples over classes. In the rest of the paper we use the terms *classes* and *categories* interchangeably.

AG's news corpus (AGNC). The news articles corpus was introduced by Antonio Gulli's group[1]. We took 4 largest classes from the corpus and used the title, and description fields as the article's text, similarly as in Zhang et al. (2015). We used weeks as time segments (Appendix. Figure 3 b)).

Multi-Domain Sentiment Dataset (MDSD). Dataset consist of product reviews taken from Amazon.com and was introduced in Blitzer et al. (2007) for solving sentiment classification task. We chose an updated version of the dataset avaliable on the web[2], provided by the same authors. Instead of trying to solve sentiment classification task, we used product type information as a target class for the product review. We used months as time segments (Appendix. Figure 3 c)).

Text Classification Dataset (TCD). Our private real-world dataset for text classification. We used months as time segments (Figure 1).

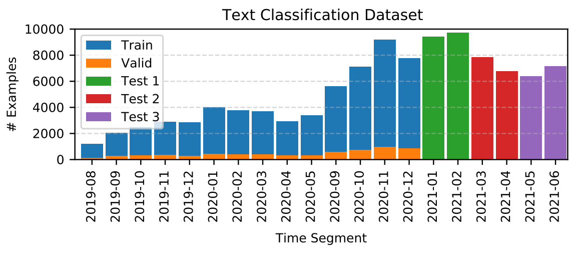

Figure 1: Number of examples per time segment and data split information for Text Classification Dataset. Test 1, 2 and 3 are subsets of the full test set.

2.2 Sampling Methods

We used 8 sampling methods to subsample the dataset. Methods 1) to 7) were used as a data pre-processing step prior to model optimization while method 8) is used after each epoch as part of the model optimization procedure. In the rest of the paper, sampling methods are also called as samplers. Samplers have a hyper-parameter called *sampling ratio* which is used to determine *sample size*, where *sampling ratio* $\in [0, 1]$. When saying at random, we mean uniformly at random unless otherwise stated. A description of the sampling methods used in the experiments is as follows (the implementation for methods 1) to 7) is available in the supplementary materials).

1) Full Dataset Sampler (FDS) takes all the examples, i.e. samples the dataset with *sampling ratio* of 1.0.

2) Latest Examples Sampler (LES) samples portion of the temporally most recent examples per each category proportional to sampling ratio. In a corner case, such that when there are more examples in a time segment than it is needed to satisfy the sample size, the examples are picked at random.

3) Random Sampler (RS) samples the portion of

Dataset	AGNC	MDSD	TCD
Classes	4	25	221
Time Segments	6	35	20
Train	46111	16517	59104
Valid	5124	1833	6563
Test 1	9026	4224	19185
Test 2	8520	7294	14635
Test 3	3249	6444	13576
Test Full	20795	17962	47396

Table 1: An overview of dataset properties. AGNC, MDSD and TCD are abbreviations for AG's news corpus, Multi-Domain Sentiment Dataset and Text Classification Dataset, respectively.

[1] http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html
[2] http://www.cs.jhu.edu/~mdredze/datasets/sentiment/

examples per each category proportional to sampling ratio at random and independently of the time segments.

Samplers 4) to 7) are clustering-based approaches for subsampling the dataset inspired by previous work of Li et al. (2011); Agrawal et al. (2015); Arafat et al. (2017). However, we didn't balance the distribution of examples over classes, but kept the same ratio of examples per category as it is prior to sampling. We did this because the cited methods are not directly applicable or practical to be used with transformers-based models so we only followed the subsampling approaches which by themselves do not balance distribution. For each clustering method we used *k-means++* (Arthur and Vassilvitskii, 2007).

4) Clustering C1 samples portion of the examples per each category proportional to sampling ratio and independently of the time segments, as follows. First, it clusters the category to partition the example space. Then, at random selects from which cluster to take the example from, repeating the sampling procedure *sample size* times. After this step, the number of examples that sampler needs to pick from each cluster is known. Finally, sampler picks determined number of examples from each cluster at random. For representing the examples as a continuous vector we apply TF-IDF.

5) Clustering C2 is same as the **C1** but samples from a non-uniform distribution, by which we model the dependence of the examples on the time segment. In short, the most recent examples and clusters containing these are more likely to be picked (Appendix. A.2).

6) Clustering C3 is the same as **C1** and **7) Clustering C4** is the same as **C2** but both use DistilBERT model to calculate vector representations.

8) CRAIG is a method proposed by Mirzasoleiman et al. (2020). We used the provided implementation by the authors[3], but adjusted it to work for the transformer-based model for text classification.

2.3 Experiments

Dataset Splits. Split statistics are shown in Table 1. Additionally, we show the splits for TCD in Figure 1 with more details in Appendix Figure 3.

Neural Net Architecture. We chose DistilBERT (Sanh et al., 2020) to simulate the use of a compressed model while we want to reduce the dataset.

Metrics. For measuring the model performance we used f1 micro score. Training times are measured against a wall clock, while we treat a total training time as a sum of the sampling time and the training time without evaluation on the validation set.

Training and Evaluation Procedure. For each sampler we trained the model for 10 epochs on the sampled training set and picked the model that achieved the best f-score on the validation set. Then, for each sampler, we evaluated the selected model on the full test set as well as on 3 test subsets. We get the 3 subsets by dividing the test set over time into 3 buckets, the main goal being to additionally test model performance for examples that are near/far in time from the train set. Each model starts from the same initial weights and uses the same hyperparameters (Appendix. A.3). What changes across the experiments is the training subset provided by each of the samplers.

3 Results

We report the mean and standard deviation over sampler's f1 scores on the full test set (Figure 2 a)) as well as the training times (Figure 2 b)), both relative to the full dataset sampler. Exact values are shown in Table 2, with additional information for the 3 test subsets. Detailed results with actual training times can be found in the supplementary materials. Secondly, we report results for the top 4 samplers w.r.t. f1 score but with condition of 0.7, 0.5 and 0.3 on the maximum training time (Table 3), where measures are relative to the full dataset sampler.

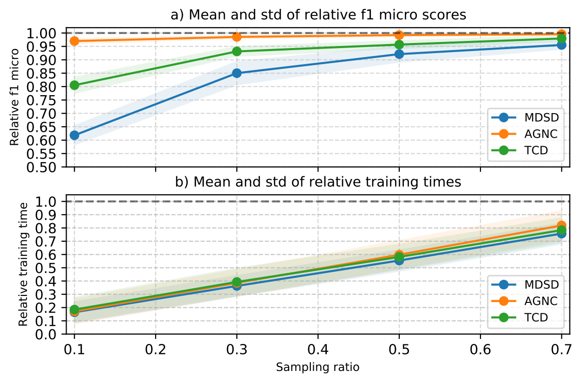

Figure 2: Samplers average f1 scores on full test set and training times, both relative to the full dataset sampler.

4 Discussion

As seen in Figure 2, on average and relative to the full dataset sampler, f-score and training time

[3]https://github.com/baharanm/craig

Samp. Ratio	Dataset	Rel. F1				Rel. Train Time
		Full	Test 1	Test 2	Test 3	
		mean ± std				mean ± std
0.3	AGNC	0.99 ± 0.01	0.98 ± 0.01	0.99 ± 0.01	0.99 ± 0.01	0.39 ± 0.10
	TCD	0.93 ± 0.02	0.94 ± 0.02	0.93 ± 0.02	0.93 ± 0.02	0.39 ± 0.10
	MDSD	0.85 ± 0.04	$\mathbf{0.88 \pm 0.03}$	$\mathbf{0.84 \pm 0.05}$	$\mathbf{0.84 \pm 0.05}$	0.36 ± 0.08
0.1	AGNC	$\mathbf{0.97 \pm 0.01}$	0.97 ± 0.01	0.97 ± 0.01	0.97 ± 0.01	0.17 ± 0.09
	TCD	$\mathbf{0.81 \pm 0.03}$	0.81 ± 0.03	0.80 ± 0.03	0.80 ± 0.03	0.18 ± 0.09
	MDSD	$\mathbf{0.62 \pm 0.03}$	$\mathbf{0.69 \pm 0.03}$	$\mathbf{0.57 \pm 0.04}$	$\mathbf{0.62 \pm 0.04}$	0.16 ± 0.08

Table 2: Training time and f-score on test sets, averaged over all samplers. Measures are relative to the full dataset sampler. Results in bold indicate when a significant drop in f-score happened w.r.t. sampling ratio.

Max Rel. Train Time	Sampler	Samp. ratio	Rel. F1	Rel. Train Time	Rel. Loss
			mean ± std	mean ± std	mean ± std
1.0	Full data	1.0	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
0.7	CRAIG	0.3	$\mathbf{0.97 \pm 0.02}$	0.61 ± 0.03	$\mathbf{0.99 \pm 0.04}$
	Clustering C2	0.5	0.96 ± 0.03	0.56 ± 0.01	1.18 ± 0.21
	Clustering C4	0.5	0.96 ± 0.03	0.58 ± 0.02	1.18 ± 0.18
	Latest Examples	0.5	0.95 ± 0.02	$\mathbf{0.50 \pm 0.02}$	1.19 ± 0.19
0.5	Latest Examples	0.3	$\mathbf{0.93 \pm 0.04}$	$\mathbf{0.31 \pm 0.01}$	1.27 ± 0.19
	Random	0.3	0.92 ± 0.05	$\mathbf{0.31 \pm 0.01}$	1.26 ± 0.21
	Clustering C4	0.3	0.92 ± 0.05	0.37 ± 0.01	$\mathbf{1.18 \pm 0.06}$
	Clustering C2	0.3	0.92 ± 0.05	0.36 ± 0.01	1.29 ± 0.21
0.3	Latest Examples	0.1	$\mathbf{0.82 \pm 0.11}$	$\mathbf{0.10 \pm 0.00}$	$\mathbf{1.50 \pm 0.12}$
	Clustering C4	0.1	$\mathbf{0.82 \pm 0.11}$	0.17 ± 0.01	1.63 ± 0.45
	Random	0.1	0.81 ± 0.13	$\mathbf{0.10 \pm 0.00}$	1.55 ± 0.27
	Clustering C1	0.1	0.81 ± 0.12	0.15 ± 0.01	1.52 ± 0.18

Table 3: Results of top 4 samplers over datasets w.r.t. f-score per condition on the maximum train time. Measures are relative to the full dataset sampler. Results in bold indicate the best score per maximum train time condition.

decline almost linearly by reducing the sampling ratio from 0.7 up to 0.3. Reducing the sampling ratio below 0.3, causes a significant non-linear drop in f-score. More precisely, focusing on the sampling ratio interval from 0.3 to 0.1, we have a drop on average of 0.2, 0.12 and 0.23 points in relative f-score for all the datasets, respectively. We assume that after discarding certain portion of the examples, we can not reduce it more without removing good representative examples from each category, i.e. there are less redundant examples to discard. Additionally, we measured relative f-score on each test subset to see if and when the drop in performance might happen, as this simulates use cases where new data arrives over time. On MDSD dataset we observed that performance drops as we try to predict further in time, when sampling ratio is 0.3 or lower (Table 2). Specifically, f-score drops from 0.69 to 0.57 and then raise to 0.62 as we go through Test 1, 2 and 3, respectively. This is likely related to the number of classes and examples per class, we leave it for further investigation.

We further discuss the effect of conditioning on the maximum allowed training time (Table 3). We chose values for maximum relative training time (Table 3, column 1) based on the Figure 2, with intention to cover the different situations regarding picked sampling ratio. With a condition of a maximum relative training time of 0.7, CRAIG achieves the best results, on average, with decrease of 0.03 points in f1 score while speeding up the training time by 63.93%, relative to the score and time obtained by using the full dataset. As we reduce the condition value below 0.5 we are left only with data preprocessing samplers. In this case, the top performing sampler w.r.t. f-score is LES. Interestingly, LES was ranked in top 4 samplers for each condition on the maximum relative training time. We assume this might be the case because we sorted the data by their timestamp and partitioned into time segments. It might be that the most recent time segments contain more representative examples for the test set as they are closer in time. Next, we compare clustering samplers. We observed that C2 and C4, the ones which apply bigger weights for more recent examples while sampling, outperformed C1 and C3, the ones that use uniform weights. This aligns with our previous assumption that the recent examples give more information for predicting on the future time segments. Additionally, C4, on average, outperformed C2, which may indicate that clustering transformer embeddings instead of using

TF-IDF vectors might provide more representative examples for model training.

Finally, for the completeness of comparison, we tested the samplers on the same datasets but without preserving temporal information. Only samplers which do not require temporal information to work were taken into account. Again, we obtained top results by using CRAIG sampler (Appendix Table 6). The samplers, on average, performed similarly w.r.t sampling ratios as on datasets with temporal information (Appendix Figure 4 and Table 7). The samplers, on average, achieved slightly better results on datasets without temporal information (Appendix Table 6). Interestingly, after setting the time condition equal to 0.7 or less, samplers are not performing better than random sampler besides CRAIG, hence we leave the topic for future investigation. As datasets without temporal information are not our primary focus, we include additional information in Appendix C about the results and the methodology used.

Conclusion. Fine-tuning a compressed transformer model can still take significant amount of time if the dataset is large. We empirically compared 8 sampling methods for reducing train set on 3 datasets and reported trade-offs between f1 score and training time.

References

Astha Agrawal, Herna L. Viktor, and Eric Paquet. 2015. SCUT: Multi-class imbalanced data classification using SMOTE and cluster-based undersampling. In *2015 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K)*, volume 01, pages 226–234.

A. Angelova, Y. Abu-Mostafam, and P. Perona. 2005. Pruning training sets for learning of object categories. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 494–501 vol. 1. ISSN: 1063-6919.

Md. Yasir Arafat, Sabera Hoque, and Dewan Md. Farid. 2017. Cluster-based under-sampling with random forest for multi-class imbalanced classification. In *2017 11th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, pages 1–6. ISSN: 2573-3214.

David Arthur and Sergei Vassilvitskii. 2007. k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '07, pages 1027–1035, USA. Society for Industrial and Applied Mathematics.

Olivier Bachem, Mario Lucic, and Andreas Krause. 2017. Practical Coreset Constructions for Machine Learning. *arXiv:1703.06476 [stat]*. ArXiv: 1703.06476.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447, Prague, Czech Republic. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, pages 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota.

Manish Gupta and Puneet Agrawal. 2021. Compression of Deep Learning Models for Text: A Survey. *arXiv:2008.05221 [cs]*. ArXiv: 2008.05221.

Agata Lapedriza, Hamed Pirsiavash, Zoya Bylinskii, and Antonio Torralba. 2013. Are all training examples equally valuable? *arXiv:1311.6510 [cs, stat]*. ArXiv: 1311.6510.

Shoushan Li, Guodong Zhou, Zhongqing Wang, Sophia Lee, and Rangyang Wang. 2011. Imbalanced sentiment classification. pages 2469–2472.

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. 2020. Coresets for Data-efficient Training of Machine Learning Models. *arXiv:1906.01827 [cs, stat]*. ArXiv: 1906.01827.

J. Arturo Olvera-López, J. Ariel Carrasco-Ochoa, J. Francisco Martínez-Trinidad, and Josef Kittler. 2010. A review of instance selection methods. *Artificial Intelligence Review*, 34(2):133–143.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv:1910.01108 [cs]*. ArXiv: 1910.01108.

Ilia Sucholutsky and Matthias Schonlau. 2020. Soft-Label Dataset Distillation and Text Dataset Distillation. *arXiv:1910.02551 [cs, stat]*. ArXiv: 1910.02551.

Ivor W. Tsang, James T. Kwok, and Pak-Ming Cheung. 2005. Core Vector Machines: Fast SVM Training on Very Large Data Sets. *Journal of Machine Learning Research*, 6(13):363–392.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. *Advances in Neural Information Processing Systems*, 32.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros. 2020. Dataset Distillation. *arXiv:1811.10959 [cs, stat]*. ArXiv: 1811.10959.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 649–657, Cambridge, MA, USA. MIT Press.

A Additional Methods Details

A.1 Dataset Splits

We created training, validation and test sets for each of the datasets described in Subsection 2.1. First, we created full test set by taking 6 most recent time segments. Then, in order to get test sets 1, 2 and 3, we divided the test set over time into 3 buckets, each containing two time segments. Thirdly, in train set are all examples from time segments older than test set 1. Finally, we created validation set by sampling 10% of the train set while keeping example ratio per category the same between the splits. The splits are fixed and do not change across experiments. Data splits for each dataset are shown in Figure 3 and split statistics are given in Table 1.

A.2 Sampling Distribution Calculation

We calculated the distributions by using weights w_i assigned to each example e_i where $i \in 1, 2, ..., |TrainSet|$. Each dataset has n train time segments T, known after segmentation and dataset split is done (Figure 3). Each time segment T has index j, where the most recent segment in time has index $j = 1$, the second most recent has index $j = 2$ and so on up to the last index $j = n$. Every example e_i belongs to one of the time segments T_j so we define membership function to get index of the segment in which example belongs to, as follows:

$$\mu_T(e_i) = \begin{cases} j & \text{if } e_i \in T_j \\ 0, & \text{otherwise} \end{cases} , j \in \{1, ..., n\} \quad (1)$$

Note that $\mu_T(e_i)$ will never be 0 as every example belongs to one of the segments. We chose linear decay function f_w to assign weight w_i to example e_i w.r.t. index j of the segment T_j to which the example e_i belongs. Function f_w is defined as follows:

$$w_i = f_w(e_i) = \frac{1}{\mu_T(e_i)} \quad (2)$$

where the weight w_i has the highest value for examples from the most recent segments and decreases linearly as the segments get older. After performing clustering we obtained m clusters. For a cluster weight c_k, we averaged weights w_i of all examples e_i which belong to the cluster k. We did this for each cluster. To determine distribution for sampling which cluster to take the example from, we calculated cluster probability p_k as follows:

$$p_k = \frac{c_k}{\sum_{l=1}^{m} c_l} , k \in \{1, ..., m\} \quad (3)$$

$$\sum_{k=1}^{m} p_k = 1 \quad (4)$$

To determine distribution for sampling examples within cluster C_k, we calculate example probability p'_i per cluster C_k, $k = 1, ..., m$, as follows:

$$p'_i = \frac{w_i}{\sum_{l=1}^{|S_k|} w_l} , l, i \in S_k \quad (5)$$

$$\sum_{i=1}^{|S_k|} p'_i = 1 \quad (6)$$

where S_k is a set of indexes of examples e_i that belongs to a cluster C_k.

A.3 Neural Network Training Info

Transformer's maximum sequence length used was 512 and we used batch size of 32. We used Adam optimizer with initial learning rate of $3e^{-5}$.

B Additional Discussion Remarks

B.1 CRAIG Remarks

In Table 5, we ranked the samplers w.r.t. relative f-score over the datasets, setting no condition on max rel. time. Top f-score performance on average is achieved by CRAIG, but if we look at the average and standard deviation of the relative training time we see that by sampling the data with ratio of 0.7 the model training was slower than on the full dataset. This is because the time needed for selecting the subsets creates the time overhead if sampling ratio is not small enough.

C Case Without Temporal Information

C.1 Dataset Splits

We created datasets without temporal information as follows. For TCD and MDSD we took splits with temporal information, merged them and discarded the time information. Subsequently, we shuffled the data and randomly split it in 8:1:1 ratio to create the training, validation and test sets. For AGNC we took the same training and test splits as in Zhang et al. (2015), and did not discard articles that do not have timestamp. Hence we could have

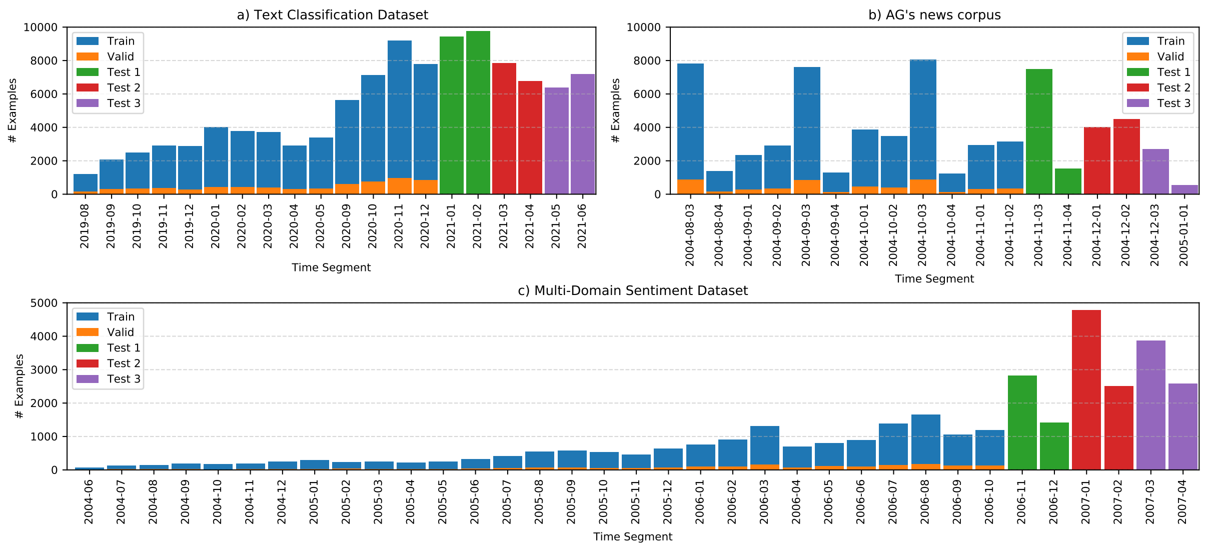

Figure 3: Number of examples per time segments together with dataset split information for each used dataset.

Samp. Ratio	Dataset	Rel. F1 Full	Rel. F1 Test 1	Rel. F1 Test 2	Rel. F1 Test 3	Rel. Train Time
		mean ± std				mean ± std
1.0	Every	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
0.7	AGNC	1.00 ± 0.00	0.99 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	0.82 ± 0.11
	TCD	0.98 ± 0.01	0.98 ± 0.01	0.98 ± 0.02	0.98 ± 0.01	0.78 ± 0.09
	MDSD	0.96 ± 0.02	0.96 ± 0.01	0.96 ± 0.02	0.95 ± 0.02	0.76 ± 0.08
0.5	AGNC	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.60 ± 0.11
	TCD	0.96 ± 0.01	0.96 ± 0.01	0.96 ± 0.02	0.96 ± 0.02	0.58 ± 0.09
	MDSD	0.92 ± 0.03	0.93 ± 0.03	0.92 ± 0.03	0.92 ± 0.03	0.55 ± 0.08
0.3	AGNC	0.99 ± 0.01	0.98 ± 0.01	0.99 ± 0.01	0.99 ± 0.01	0.39 ± 0.10
	TCD	0.93 ± 0.02	0.94 ± 0.02	0.93 ± 0.02	0.93 ± 0.02	0.39 ± 0.10
	MDSD	0.85 ± 0.04	$\mathbf{0.88 \pm 0.03}$	$\mathbf{0.84 \pm 0.05}$	$\mathbf{0.84 \pm 0.05}$	0.36 ± 0.08
0.1	AGNC	$\mathbf{0.97 \pm 0.01}$	0.97 ± 0.01	0.97 ± 0.01	0.97 ± 0.01	0.17 ± 0.09
	TCD	$\mathbf{0.81 \pm 0.03}$	0.81 ± 0.03	0.80 ± 0.03	0.80 ± 0.03	0.18 ± 0.09
	MDSD	$\mathbf{0.62 \pm 0.03}$	$\mathbf{0.69 \pm 0.03}$	$\mathbf{0.57 \pm 0.04}$	$\mathbf{0.62 \pm 0.04}$	0.16 ± 0.08

Table 4: Average f1 scores and training times on test sets for each sampler, both relative to the full dataset sampler. Results in bold signal when a significant drop in f-score happened w.r.t. sampling ratio.

one dataset that is often used as a benchmark for text classification. To create AGNC validation set we randomly sampled 10% of the AGNC training set. We kept the same ratio of examples per categories across splits for each dataset. The splits statistics are presented in Table 8.

C.2 Results

We report the mean and standard deviation over samplers f1 scores on the test set (Figure 4 a)) and the training times (Figure 4 b)), both relative to the full dataset sampler. Exact values are shown in Table 7. Detailed results with measured training times are in the supplementary materials. Next, we report results for the top 4 samplers w.r.t. f1 score but with condition of 0.7, 0.5 and 0.3 on the maximum training time (Table 6), where measures are relative to the full dataset sampler.

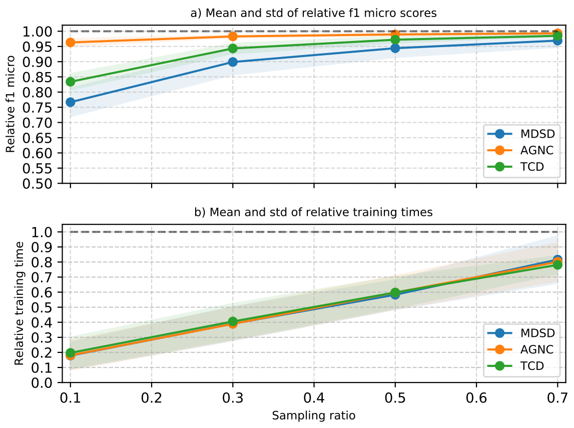

Figure 4: Samplers average f1 scores on test set and training times on datasets without time dimension, both relative to the full dataset sampler.

Max Rel. Train Time	Sampler	Samp. ratio	Rel. F1	Rel. Train Time	Rel. Loss
			mean ± std	mean ± std	mean ± std
1.0	Full data	1.0	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
inf	CRAIG	0.7	**1.00 ± 0.00**	1.00 ± 0.05	**1.01 ± 0.08**
inf	CRAIG	0.5	0.99 ± 0.00	0.79 ± 0.04	1.04 ± 0.12
inf	Latest Examples	0.7	0.98 ± 0.01	**0.70 ± 0.03**	1.05 ± 0.11
inf	Clustering C4	0.7	0.98 ± 0.01	0.77 ± 0.01	1.10 ± 0.17
0.7	CRAIG	0.3	**0.97 ± 0.02**	0.61 ± 0.03	**0.99 ± 0.04**
0.7	Clustering C2	0.5	0.96 ± 0.03	0.56 ± 0.01	1.18 ± 0.21
0.7	Clustering C4	0.5	0.96 ± 0.03	0.58 ± 0.02	1.18 ± 0.18
0.7	Latest Examples	0.5	0.95 ± 0.02	**0.50 ± 0.02**	1.19 ± 0.19
0.5	Latest Examples	0.3	**0.93 ± 0.04**	**0.31 ± 0.01**	1.27 ± 0.19
0.5	Random	0.3	0.92 ± 0.05	**0.31 ± 0.01**	1.26 ± 0.21
0.5	Clustering C4	0.3	0.92 ± 0.05	0.37 ± 0.01	**1.18 ± 0.06**
0.5	Clustering C2	0.3	0.92 ± 0.05	0.36 ± 0.01	1.29 ± 0.21
0.3	Latest Examples	0.1	**0.82 ± 0.11**	**0.10 ± 0.00**	**1.50 ± 0.12**
0.3	Clustering C4	0.1	**0.82 ± 0.11**	0.17 ± 0.01	1.63 ± 0.45
0.3	Random	0.1	0.81 ± 0.13	**0.10 ± 0.00**	1.55 ± 0.27
0.3	Clustering C1	0.1	0.81 ± 0.12	0.15 ± 0.01	1.52 ± 0.18

Table 5: Results of top 4 samplers on average over datasets w.r.t. relative f1 micro scores with condition on the maximum relative training time. Results in bold indicate the best scores for specific condition on the maximum training time. inf condition represents no limit on the maximum training time.

Max Rel. Train Time	Sampler	Samp. ratio	Rel. F1	Rel. Train Time	Rel. Loss
			mean ± std	mean ± std	mean ± std
1.0	Full data	1.0	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
inf	CRAIG	0.7	**0.99 ± 0.01**	0.95 ± 0.09	1.09 ± 0.16
inf	CRAIG	0.5	0.99 ± 0.00	0.77 ± 0.01	**0.92 ± 0.36**
inf	Clustering C3	0.7	0.98 ± 0.01	0.77 ± 0.01	1.02 ± 0.40
inf	Clustering C1	0.7	0.98 ± 0.01	**0.75 ± 0.03**	0.94 ± 0.35
0.7	CRAIG	0.3	**0.98 ± 0.01**	0.59 ± 0.02	1.01 ± 0.10
0.7	Random	0.5	0.97 ± 0.02	**0.50 ± 0.01**	**0.98 ± 0.36**
0.7	Clustering C3	0.5	0.97 ± 0.02	0.55 ± 0.00	1.34 ± 3.52
0.7	Clustering C1	0.5	0.96 ± 0.02	0.55 ± 0.01	1.14 ± 0.49
0.5	Random	0.3	**0.95 ± 0.04**	0.30 ± 0.00	**1.07 ± 0.41**
0.5	Clustering C3	0.3	0.94 ± 0.04	0.35 ± 0.00	1.10 ± 0.43
0.5	Clustering C1	0.3	0.94 ± 0.03	0.34 ± 0.02	1.08 ± 0.41
0.5	Random	0.1	0.87 ± 0.07	**0.10 ± 0.00**	1.40 ± 0.59
0.3	Random	0.1	**0.87 ± 0.07**	**0.10 ± 0.00**	1.40 ± 0.59
0.3	Clustering C3	0.1	0.86 ± 0.07	0.15 ± 0.00	1.40 ± 0.58
0.3	Clustering C1	0.1	0.86 ± 0.07	0.14 ± 0.02	**1.34 ± 0.53**

Table 6: Results of top 4 samplers on average over datasets w.r.t. relative f1 micro scores with condition on the maximum relative training time. Results in bold indicate the best scores for specific condition on the maximum training time. inf condition represents no limit on the maximum training time. Datasets have no time dimension.

Samp. Ratio	**Dataset**	**Rel. F1**	**Rel. Train Time**
		Test	
		mean ± std	**mean ± std**
1.0	Every	1.00 ± 0.00	1.00 ± 0.00
0.7	AGNC	0.99 ± 0.00	0.80 ± 0.13
	TCD	0.98 ± 0.00	0.78 ± 0.06
	MDSD	0.97 ± 0.02	0.82 ± 0.16
0.5	AGNC	0.99 ± 0.00	0.60 ± 0.11
	TCD	0.97 ± 0.01	0.60 ± 0.10
	MDSD	0.94 ± 0.03	0.58 ± 0.10
0.3	AGNC	0.98 ± 0.00	0.39 ± 0.11
	TCD	0.94 ± 0.02	0.41 ± 0.12
	MDSD	0.90 ± 0.04	0.39 ± 0.11
0.1	AGNC	0.96 ± 0.01	0.18 ± 0.10
	TCD	0.83 ± 0.03	0.20 ± 0.10
	MDSD	0.77 ± 0.05	0.18 ± 0.09

Table 7: Average f1 scores and training times on test sets for each sampler, both relative to the full dataset sampler. Datasets have no time dimension.

Dataset	**AGNC**	**MDSD**	**TCD**
Classes	4	25	221
Train	108000	29049	90450
Valid	12000	3631	11306
Test	7600	3632	11307

Table 8: An overview of dataset properties where time dimension is excluded. AGNC, MDSD and TCD are abbreviations for AG's news corpus, Multi-Domain Sentiment Dataset and Text Classification Dataset, respectively.