Scientific Research Publishing

International Journal of Intelligence Science, 2027, 17(1), 13-48
https://www.scirp.org/journal/ijis
ISSN Online: 2163-0356
ISSN Print: 2163-0283

The Cost of Accuracy in Text Classification: Tradeoffs in Data Representation and Model Design for Classical ML, Neural Network, and Transformer-Based Models

Peter J. Worth Jr.

Department of Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL, USA
Email: pworth2022@fau.edu

How to cite this paper: Worth Jr., P.J. (2027) The Cost of Accuracy in Text Classification: Tradeoffs in Data Representation and Model Design for Classical ML, Neural Network, and Transformer-Based Models. *International Journal of Intelligence Science*, **17**, 13-48.
https://doi.org/10.4236/ijis.2027.171002

Received: October 24, 2026
Accepted: November 23, 2026
Published: November 26, 2026

Copyright © 2027 by author(s) and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/



Open Access

Abstract

Fine-tuned transformers are the default choice for text classification, yet the accuracy they add over cheaper alternatives, and the compute they spend to add it, are seldom reported side by side. We measure both across 3725 training runs of eight classifiers in three families on eight single- and multi-label corpora. The families are linear models (logistic regression, linear SVM, naive Bayes) over sparse, latent-semantic and pooled-embedding features; CNN, LSTM and attention networks over static word embeddings; and seven fine-tuned transformers, from DistilBERT to Llama-3.2-1B and DeepSeek-R1-Distill-Qwen-1.5B. Comparing the best test-set run of each family on each corpus, fine-tuned transformers lead everywhere, but by a mean of only 3.2 Macro-F1 points over the best linear configuration and 6.7 points over the best static-embedding network. Meanwhile, a typical transformer run trains 75 times longer than a typical linear run, and 32 times longer than a typical static-embedding run (127 times its GPU-time). The premium depends on the task. On the four single-label corpora, a linear model within 0.5 - 2.7 points of the best transformer needs under 0.5% of its classifier-fitting time; on the four multi-label, many-class corpora, the transformer's 3.7 - 7.0-point lead over the best linear configuration has no linear substitute. Within the transformer family, the two billion-parameter-scale decoders cost about five times as much to fine-tune as base-size encoders for no consistent gain. Word-class embeddings help static-embedding networks in 80 of 129 paired runs, but not consistently across corpora. For their token-level analogue in fine-tuned transformers, no statistically detectable systematic effect was found in these runs. The savings are in classifier fitting only: the strongest linear configurations use frozen transformer embeddings, which also need transformer inference on new doc-

uments, so deployment savings are unmeasured. We release the run-level results and the analysis scripts.

Keywords

Text Classification, Computational Cost, Fine-Tuning, Transformers, Large Language Models, Word Embeddings, Word-Class Embeddings, Language Representation, Pareto Efficiency, Green AI

1. Introduction

Text classification is among the most widely deployed tasks in natural language processing: topic routing, spam and abuse filtering, clinical coding, sentiment monitoring and threat triage all reduce to assigning labels to documents. Since BERT [1], the standard recipe has been to fine-tune a pretrained transformer [2]-[4], and the models used for that recipe have grown from roughly a hundred million parameters to billions [5] [6]. Benchmarks and surveys of text classification report how much more accurate these models are [7]-[9]; they rarely report what that accuracy costs to obtain. Calls to report compute alongside accuracy are by now several years old [10]-[13], and the energy and dollar cost of large models has been documented at length [14] [15]. What a practitioner choosing a classifier still lacks is a direct answer to two questions about their own task: *how many points of F*1 *does a fine-tuned transformer add over the cheapest competitive alternative, and how much additional training compute does each of those points require*?

There are good reasons to expect the answer to vary. Linear models over sparse features remain strong text classifiers [16]-[18], simple pooled word embeddings are competitive with far more elaborate encoders [19], and a wide bag-of-words perceptron matches or beats graph-based classifiers on standard benchmarks, although fine-tuned transformers still lead [20]. At the same time, fine-tuned small models still outperform prompted generative LLMs on supervised classification, whether the LLMs are used zero-shot or with in-context examples [21] [22], so fine-tuning is not going away. A useful study therefore has to place classical, static-embedding and transformer classifiers on a common footing and measure effectiveness and cost on the same runs.

This article does so. We build on Layer Cake, an open benchmarking framework that runs three families of classifiers (linear models, neural networks over static word embeddings, and fine-tuned transformers) over eight single- and multi-label corpora, logging Macro- and Micro-F1, training time and hardware for every run [23]. Here we re-analyse its 3725 runs as a cost-effectiveness study. We separate the *peak* score a family can reach on a corpus from its *typical* score across configurations. We test family differences with paired non-parametric tests across corpora [24], express cost both as wall-clock training time and as accelerator-time, and trace, for each corpus, the Pareto frontier of score against training

time and the cheapest run within a given tolerance of the best. We also ask two narrower questions the earlier study raised but did not settle: whether larger transformers are worth their cost, and whether supervised class embeddings help the models they are attached to.

Five findings stand out.

1) **The transformer premium is real but small on average.** At peak, fine-tuned transformers beat the best linear configuration on all eight corpora, but by a mean of 3.2 Macro-F1 points (range 0.1 - 7.0), and the best static-embedding network by 6.7 points (0.4 - 16.8). A typical transformer run takes 75 times as long to train as a typical linear run and 32 times as long as a typical static-embedding run (127 times the GPU-time; geometric means).

2) **The premium depends on the task.** On the four single-label corpora, the best linear configuration trails by 0.1 - 2.3 points, and a linear run within 0.5 - 2.7 points of the best transformer needs 0.2% - 0.5% of its classifier-fitting time. On the four multi-label corpora, which have 23 - 115 classes, the transformer's lead of 3.7 - 7.0 points cannot be obtained without a transformer.

3) **Linear models over frozen transformer embeddings beat static-embedding networks.** The best linear configuration outperforms the best CNN, LSTM or attention network over static word vectors on every corpus (mean +3.5 Macro-F1), and no static-embedding run lies on any Macro-F1 Pareto frontier. On six of the eight corpora, the winning linear configuration uses mean-pooled embeddings from a frozen transformer, whose extraction is not included in its training time and which require a transformer forward pass for every new document.

4) **Larger transformers are not worth their cost here.** DeepSeek-R1-Distill-Qwen-1.5B has the best average rank among the seven fine-tuned models but takes 6.6 times as long to train as BERT; Llama-3.2-1B ranks sixth of seven at 5.1 times. Against the best base-size encoder on each corpus, the better of the two large models changes Macro-F1 by a median of −0.4 points at 4.9 times the training time.

5) **In the neural models, no class-embedding effect is significant across corpora.** Appending word-class embeddings (WCEs) [25] to static word vectors raises Macro-F1 in 80 of 129 paired runs (median +0.8), but the per-corpus mean change is positive on five corpora and negative on three ($p = 0.31$), with the gains concentrated on arXiv and Reuters-21578. For the token-level analogue (TCEs) fused with a fine-tuned transformer, no statistically detectable systematic effect was found in these runs (median 0.0 across 79 pairs; $p = 0.55$).

All winners in these findings are selected on test-set scores, as the run logs hold no validation scores; they are descriptive test-set maxima (Section 3.6).

Contributions.

i) A controlled comparison of eight classifiers in three families on eight corpora in which effectiveness and training cost are measured on the same runs. It comes with a reporting protocol (peak and typical scores, paired tests, accelerator-time, Pareto frontiers and tolerance tables) that keeps the two quantities apart.

ii) Evidence that the transformer premium is concentrated in multi-label,

many-class tasks, with the cheapest run within a given tolerance of the best identified for every corpus.

iii) Evidence that linear models over pooled frozen embeddings dominate static-embedding networks and populate most of the low-cost frontier, with an explicit statement of what that comparison leaves out.

iv) Paired analyses showing that billion-parameter-scale decoders do not repay their cost over base encoders on these tasks, and that, in static networks and in fine-tuned transformers, no effect of class embeddings is significant across corpora.

v) The run-level results and every analysis script, so that each number and figure can be regenerated.

Relation to the earlier version. This article revises and extends [23]. The experimental runs are the same, but the analysis is new and is computed from the run-level records, which reproduce every summary statistic published earlier. We replace pooled averages over unequal numbers of configurations with per-corpus peak and typical comparisons. We add significance tests, accelerator-time, Pareto, tolerance, model-size and paired class-embedding analyses. Re-deriving the earlier paper's results led to three substantive corrections:

- Its headline comparison with classical models (a 12.73% Macro-F1 advantage for transformers) contrasted the best transformer with the *average* classical configuration. Against the best classical configuration, the advantage is 4.2% (3.2 points), and we report both.
- Its report that token-class embeddings degraded transformer performance is not supported by the logged runs, in which no statistically detectable systematic effect was found.
- Its descriptions of the hardware, language-model checkpoints and datasets contained errors, which are corrected here from the run logs and the model configurations.

The rest of the article is organised as follows. Section 2 reviews related work. Section 3 describes the framework, data, models, hardware and analysis protocol. Section 4 reports results, Section 5 discusses practical implications and limitations, and Section 6 concludes.

2. Background and Related Work

2.1. From Sparse Vectors to Contextual Representations

Early statistical text classifiers represented a document as a sparse vector over the vocabulary, weighted by term frequency and inverse document frequency [26] [27]. Because such vectors have tens of thousands of dimensions, they were often compressed, most commonly by latent semantic analysis (LSA), a truncated singular value decomposition of the term-document matrix [28]. Dense learned word vectors appeared with neural language models [29] [30] and became ubiquitous with word2vec [31] [32], GloVe [33] and fastText [34] [35]. All of these rest on the distributional hypothesis, that words occurring in similar contexts have simi-

lar meanings [36] [37], and all assign one vector per word type, so they cannot represent the sense a word takes in context [38].

Contextual representations remove that limitation. ELMo derived token vectors from a bidirectional LSTM language model [39]. The transformer [2], building on attention for sequence transduction [40], made it practical to pretrain deep contextual encoders such as BERT [1], RoBERTa [41] and XLNet [42], and decoder-only models such as GPT-2 [43]. Distillation produces smaller variants such as DistilBERT [44] [45]. Current open decoder models, including the Llama 3 family [5] and models distilled from DeepSeek-R1 into Qwen2.5 bases [6] [46], are larger again. All of these models operate on subword vocabularies produced by WordPiece [47] [48], byte-pair encoding [49] [50] or SentencePiece [51]. The size of that vocabulary fixes the size of the model's input embedding matrix, a point that matters for the cost comparisons below.

2.2. Classifiers for Text

Linear classifiers over sparse features (naive Bayes [52], support vector machines [16] and regularised logistic regression [53]) defined the state of the art for two decades [54]. Neural classifiers over static embeddings followed: convolutional networks [55] [56], recurrent networks built on LSTM units [57], and attention-based recurrent models [58]. Fine-tuning a pretrained language model end to end [1] [3] [4] then became standard. Surveys trace this progression [7]-[9] [59], and recent studies revisit text classification in the age of large language models [60] and compare fine-tuned small models with prompted LLMs [21] [22]. These studies compare effectiveness; they do not report the training cost of each approach on the same runs.

2.3. Strong Baselines and the Cost of Accuracy

A recurring finding is that simple models are hard to beat:

- Wang and Manning showed that naive-Bayes-weighted linear models are strong baselines for topic and sentiment classification [17].
- fastText showed that a linear model over averaged word and n-gram embeddings is often on par with deep classifiers in accuracy while training orders of magnitude faster [18].
- SWEM showed that pooled word embeddings rival recurrent and convolutional encoders [19].
- Galke and Scherp showed that a wide multilayer perceptron over bag-of-words features matches or outperforms graph-based classifiers, while fine-tuned BERT and DistilBERT remain ahead [20].

Separately, a literature on the cost of NLP has made three arguments: efficiency should be reported as a first-class result [10] [11] [13]; results should be reported as a function of the computational budget spent to obtain them [12]; and task-specific models are far cheaper to run than general-purpose ones [15]. Efficiency methods for transformers have been surveyed in depth [61]. Among them, pa-

rameter-efficient fine-tuning reduces the number of trainable weights through adapters [62], prefix tuning [63], low-rank adaptation [64] and its quantised and adaptive variants [65] [66], reviewed in [67]. Our study sits between these literatures: we evaluate full fine-tuning, the most expensive common practice, against the strong cheap baselines that the first literature recommends, and we report cost in the way the second recommends.

2.4. Supervised Class Embeddings

Word-class embeddings (WCEs) represent each vocabulary term by its correlation with each class in the training set, computed from the co-occurrence of terms and labels in the training documents [25]; the code base supports several such supervised functions, including a normalised dot product, positive pointwise mutual information, information gain and χ^2. Concatenated to pretrained word vectors, they add class-specific evidence that general-purpose embeddings lack. Moreo *et al.* report consistent gains when WCEs are added to four neural architectures for topic classification [25], and Layer Cake inherits its static-embedding neural module from that work. Whether the same information helps a fine-tuned transformer, whose representations are themselves shaped by the labels during fine-tuning, has not been tested on matched runs; Section 4.6 provides that test.

3. Experimental Framework

3.1. Overview

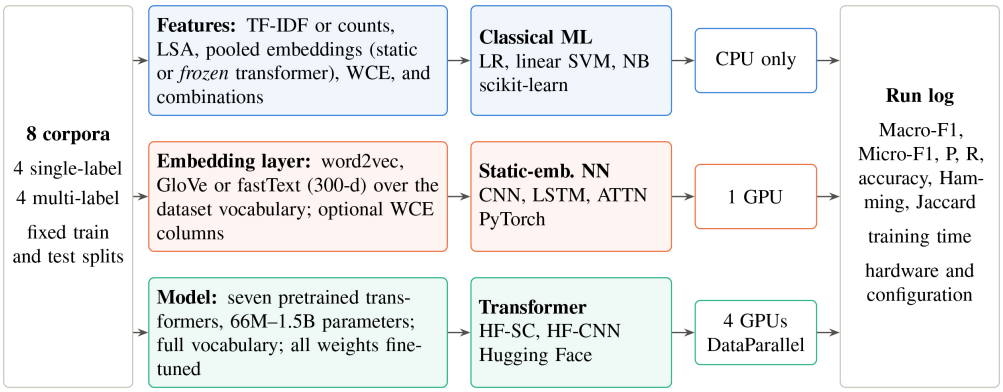

Figure 1. The three Layer Cake modules. Each corpus is processed by all three; colours mark the classifier family throughout the article. The classical module can consume embeddings from the same static and transformer language models used by the other two modules, but only as frozen, pooled features.

Layer Cake runs three modules that share datasets, splits and logging but differ in how text is represented and which hardware they use (**Figure 1**). The classical module trains scikit-learn classifiers [68]; the static-embedding module, extended from the code of [25], trains PyTorch networks [69] over pretrained word vectors; and the transformer module fine-tunes Hugging Face models [70]. Every run writes one record per metric with the corpus, classifier, language model, representation, hyperparameters, training time, operating system, CPU, GPU and memory.

All code is available at https://github.com/pworth1971/layer-cake.

3.2. Datasets

Table 1 lists the eight corpora, grouped by task type. The single-label corpora are:

- BBC News: the 1490 labelled articles distributed for the Kaggle BBC News classification task, drawn from the corpus of [71];
- 20 Newsgroups, with its standard by-date split [72];
- the IMDB sentiment corpus, with its 25,000/25,000 split [73];
- ArXiv-10: 100,000 abstracts in ten subject categories, released with Protoformer [74].

The multi-label corpora are:

- Reuters-21578, with the ModApté split [75];
- OHSUMED medical abstracts labelled with 23 MeSH disease categories [16] [76];
- a 58-category corpus of arXiv abstracts built from the public arXiv metadata [77];
- RCV1-v2, with the LYRL2004 split, in which only 23,149 of 804,414 documents are used for training [78]. RCV1-v2 defines 103 topic categories, of which the 101 that occur in this split are used.

The classical module trains on the full training split. The neural modules hold out part of it for validation and early stopping on validation Macro-F1: 20% in the static-embedding module and 17.5% in the transformer module (exact sizes in Table A3). Test sets are identical across modules.

Table 1. Corpora. Classes are those occurring in the split used. Train and test sizes are those of the classical module; the neural modules hold out part of the training split for validation.

Corpus	Domain	Classes	Documents	Train	Test
Single-label					
BBC News	news	5	1490	1229	261
20 Newsgroups	Usenet posts	20	18,846	11,314	7532
IMDB	movie reviews (sentiment)	2	50,000	25,000	25,000
ArXiv-10	paper abstracts	10	100,000	82,500	17,500
Multi-label					
Reuters-21578	newswire	115	12,902	9603	3299
OHSUMED	medical abstracts	23	34,389	24,061	10,328
arXiv	paper abstracts	58	14,500	11,962	2538
RCV1-v2	newswire	101	804,414	23,149	781,265

3.3. Representations and Language Models

Language models. Table 2 lists the ten pretrained models. The three static models supply one 300-dimensional vector per word type. The fastText vectors

used are the word-level Common Crawl vectors of [35], without subword information, so out-of-vocabulary words are treated the same way for all three static models. The seven transformer models range from 66 M to about 1.5 B parameters and use three tokenizer families.

Table 2. Pretrained language models. For static models the vocabulary column gives the number of word vectors; for transformers it gives the tokenizer vocabulary, which fixes the number of rows in the input embedding matrix. Parameter counts are approximate. Training data are as reported by the model authors.

Model	Objective	Tokenizer	Vocab.	Dim.	Pretraining data
GloVe [33]	co-occurrence factorisation	word	2.2 M	300	Common Crawl, 840 B tokens
word2vec [31] [32]	negative sampling	word	3.0 M	300	Google News, ~100 B words
fastText [35]	CBOW, position weights	word	2.0 M	300	Common Crawl, 600 B tokens
BERT (110 M) [1]	masked LM + NSP	WordPiece	30,522[a]	768	BooksCorpus, Wikipedia
DistilBERT (66 M) [45]	distillation from BERT	WordPiece	28,996/30,522[b]	768	as BERT
RoBERTa (125 M) [41]	masked LM	byte-level BPE	50,265	768	160 GB of books, web, news
XLNet (117 M) [42]	permutation LM	SentencePiece	32,000	768	books, Wikipedia, web
GPT-2 (124 M) [43]	next-token prediction	byte-level BPE	50,257	768	WebText
Llama-3.2 (1.23 B) [5] [79]	next-token + distillation	BPE (tiktoken)	128,256	2048	public data, up to 9 T tokens
DeepSeek-R1-Distill (1.5 B) [6]	SFT on 800k R1-curated samples	byte-level BPE	151,936	1536	Qwen2.5-Math-1.5B base [46]

Checkpoints: glove.840B.300d, GoogleNews-vectors-negative300, crawl-300d-2M, bert-base-uncased, distilbert-base-{cased,uncased}, roberta-base, xlnet-base-cased, gpt2, meta-llama/Llama-3.2-1B, deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B. [a]Fine-tuned runs use the uncased model, as recorded by the vocabulary size in the run log. [b]Both variants appear in the fine-tuned runs.

Preprocessing. For the static word models, text is lower-cased and stripped of punctuation and stop words. For transformer models, raw text is passed to each model's own tokenizer.

Feature sets for the classical module. We use the following notation:

- $\boldsymbol{x}_d$: the TF-IDF vector of document d (sublinear term frequency, minimum document frequency 5, as in the code base inherited from [25]);
- E_M : the matrix of embeddings that language model M assigns to the vocabulary;
- $\bar{\boldsymbol{e}}_d^M$: the pooled embedding of d . For static models it is the mean (or TF-IDF-weighted mean) of the document's word vectors; for transformers it is the mean of the frozen model's final-layer hidden states over the padded input sequence;
- W : the WCE matrix computed from the training split [25];
- $\oplus$: concatenation;
- $\Pi_{M+\mathrm{WCE}}(d) = \boldsymbol{x}_d \left[E_M \oplus W\right]$: the projection of the TF-IDF vector onto the WCE-extended embedding space.

The ten feature sets (*representation forms*) are defined in **Table 3**. All embeddings are frozen; only the linear classifier is trained. Because $\Pi_{M+\mathrm{WCE}}(d) = \boldsymbol{x}_d E_M \oplus \boldsymbol{x}_d W$, only *dot-wce* differs from its base form (*dot*) by the WCE columns $\boldsymbol{x}_d W$ alone.

The other *-wce* forms append the whole projection to their base form, adding both the unsupervised projected embedding features $\boldsymbol{x}_d E_M$ and the WCEs.

Table 3. Feature sets (representation forms) used by the classical module.

Name	Features	Dimension
vmode	$\boldsymbol{x}_d$ (TF-IDF or raw counts)	$\lvert V_D \rvert$
lsa	truncated SVD of $\boldsymbol{x}_d$	d_M
solo	$\bar{\boldsymbol{e}}_d^M$	d_M
dot	$\boldsymbol{x}_d E_M$	d_M
cat-doc	$\boldsymbol{x}_d \oplus \bar{\boldsymbol{e}}_d^M$	$\lvert V_D \rvert + d_M$
lsa-wce	$\mathrm{SVD}(\boldsymbol{x}_d) \oplus \Pi_{M+\mathrm{WCE}}(d)$	$2d_M + \lvert \mathcal{C} \rvert$
solo-wce	$\bar{\boldsymbol{e}}_d^M \oplus \Pi_{M+\mathrm{WCE}}(d)$	$2d_M + \lvert \mathcal{C} \rvert$
dot-wce	$\Pi_{M+\mathrm{WCE}}(d)$	$d_M + \lvert \mathcal{C} \rvert$
cat-wce	$\boldsymbol{x}_d \oplus \Pi_{M+\mathrm{WCE}}(d)$	$\lvert V_D \rvert + d_M + \lvert \mathcal{C} \rvert$
cat-doc-wce	$\boldsymbol{x}_d \oplus \bar{\boldsymbol{e}}_d^M \oplus \Pi_{M+\mathrm{WCE}}(d)$	$\lvert V_D \rvert + 2d_M + \lvert \mathcal{C} \rvert$

3.4. Classifiers

Classical module. Logistic regression (LR), linear SVM and multinomial naive Bayes (NB) from scikit-learn, with default hyperparameters, wrapped one-vs-rest for multi-label corpora. NB requires non-negative inputs, so it is run only on count and TF-IDF features. Each classifier is run with every compatible combination of feature set, language model and pooling, which yields 171 - 244 configurations per corpus for LR and the SVM and 18 - 38 for NB.

Static-embedding module. CNN [55], LSTM [57] and attention (ATTN) [58] classifiers, each with GloVe, word2vec and fastText inputs. The pretrained vectors are either frozen or fine-tuned, with and without WCEs, over a small grid of channel or hidden sizes (4 - 24 runs per classifier and corpus). The embedding layer contains only the words of the corpus vocabulary, plus unknown and padding tokens. WCEs (information gain, z-scored) are appended as extra columns (**Figure 2(a)**). Single-label corpora use cross-entropy loss and multi-label corpora binary cross-entropy with logits.

Transformer module. Each of the seven models is fine-tuned in full with two heads: the Hugging Face sequence-classification head (HF-SC) and a convolutional head over the final hidden states modelled on the static CNN (HF-CNN). There are 8 - 25 runs per classifier and corpus. In pilot runs, freezing lower layers reduced effectiveness, so all weights are updated. Unlike the static networks, the transformer's input embedding matrix is fixed by the pretrained model and spans its full tokenizer vocabulary (e.g. 30,522 × 768 for BERT and 128,256 × 2048 for Llama-3.2-1B), whatever the size of the corpus vocabulary (**Figure 2(b)**).

Token-class embeddings. To test whether the WCE idea carries over to transformers, the transformer module computes token-class embeddings (TCEs) over each model's subword vocabulary (normalised dot product, z-scored). Concate-

nating TCEs at the input would change the hidden width that every pretrained layer expects, so in the logged runs they are concatenated with the pooled encoder output before the classification head and fine-tuned with the rest of the model. For BERT, DistilBERT, RoBERTa, XLNet and GPT-2, every head is run with and without TCEs on every corpus, which yields the matched pairs analysed in Section 4.6.

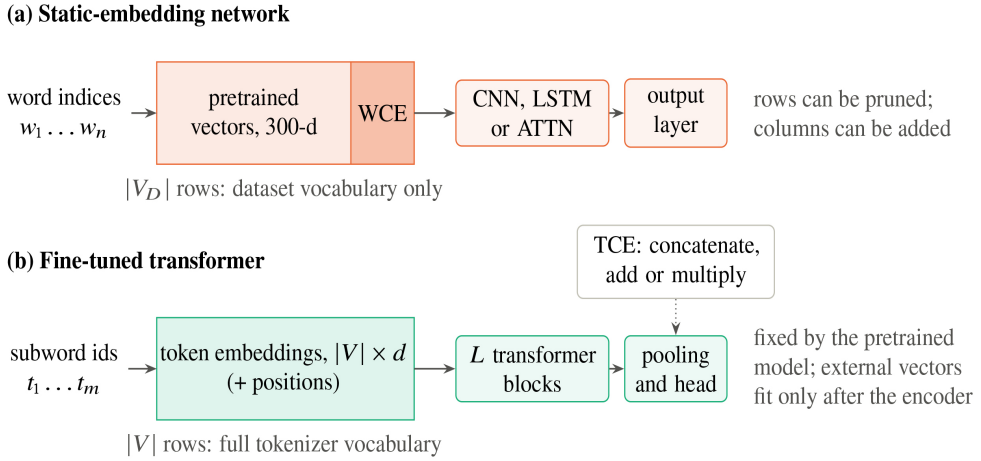

Figure 2. Where the input representation lives. (a) The static-embedding networks build their embedding layer from the corpus vocabulary and can append word-class embeddings (WCE) as extra columns. (b) A fine-tuned transformer's embedding matrix covers the model's full vocabulary, and its width is the hidden size that every pretrained layer expects, so token-class embeddings (TCE) can only be fused after the encoder.

3.5. Hardware and Cost Accounting

The run log records the machine for 89% of runs (Table 4). Every logged transformer run held four GPUs under PyTorch DataParallel, on one of three machines (NVIDIA A100, RTX A6000 or RTX 6000 Ada). Every logged static-embedding run held one RTX 6000 Ada. Classical runs used CPU only: 78% on a 64-core Linux workstation and 22% on an Apple M3 Max laptop, including all RCV1-v2 runs.

Table 4. Hardware by module, from the run log. Share of runs: share of the module's logged runs on that machine. Software: Python 3.10, scikit-learn 1.6.1, PyTorch 2.4.1, CUDA 12.4.

Module	Machine	Accelerators	Share of runs	Corpora
Classical ML	Linux workstation, 64-core CPU	none	78%	7
	Apple M3 Max laptop, 16-core CPU		22%	6
Static-embedding NN	NVIDIA RTX 6000 Ada (48 GB)	1 GPU	100%	8
Fine-tuned transformer	NVIDIA A100 PCIe (40 GB)	4 GPUs	46%	8
	NVIDIA RTX A6000 (48 GB)		35%	7
	NVIDIA RTX 6000 Ada (48 GB)		19%	8

We report two cost measures.

- *Training time* is the wall-clock time of model fitting as logged by each module, including all epochs up to early stopping for the neural models. It excludes data loading, preprocessing, tokenisation, the extraction of frozen embeddings for the classical module, and inference on the test set. Every cost ratio and saving reported in this article is therefore a classifier-fitting cost or saving; no deployment cost is measured.
- *Accelerator-time* multiplies training time by the number of accelerators the run held: four GPUs for transformers, one for static-embedding networks and none for classical models. The four-fold factor measures resource occupancy, not speed-up: the runs used PyTorch's single-process DataParallel rather than distributed data parallelism, the more efficient way to train on several GPUs [80].

Because the families ran on different machines, and the transformers on three GPU types, ratios between families combine algorithmic and hardware differences. We report them as the costs a practitioner with this equipment would face, not as hardware-normalised efficiencies.

3.6. Effectiveness Measures and Analysis Protocol

We report Macro-F1, the unweighted mean of per-class F1, and Micro-F1, F1 computed from true and false positives and false negatives pooled over classes, following [25]. Macro-F1 weights rare classes as heavily as frequent ones and is the primary measure; the neural modules also use it for early stopping. Precision, recall, accuracy, Hamming loss and Jaccard index are logged but not analysed.

Data. The analysis uses the run-level records of the original study: 3725 runs (3197 classical, 303 static-embedding, 225 transformer), each with Macro-F1 and training time. All but one also have Micro-F1. The log record of one IMDB naive Bayes run (TF-IDF features over the XLNet tokenizer) ends after its Macro-F1 entry, so the Micro-F1 analyses use 3724 runs, and **Table A5** lists 19 IMDB naive Bayes runs against 20 in **Table A4**. We aggregated the runs per corpus and classifier into the maximum, mean, median, minimum and run count. All 784 values checked match the statistics published in [23] to three decimals. Training time is available at full precision for 89% of runs and rounded to the second for the rest.

Peak and typical scores. A family's *peak* on a corpus is its best run (ties broken by lower training time), and the cost of the peak is that run's training time. A family's *typical* score is the median score of its best-by-median classifier, and its typical cost is the median training time of all its runs on the corpus. Both measures are robust to the occasional collapsed run (Section 4.2). The two answer different questions. Peak asks what a family can achieve when its configuration is chosen well; typical asks what an arbitrary configuration delivers. Because the classical module explores far more configurations than the neural ones, peak comparisons favour it and typical comparisons penalise it, and we report both.

Selection on test scores. No reported winner is chosen on validation data. The classical module has no validation split. The neural modules use theirs only for early stopping within a run, and the run log records test-set scores alone. Every selection across runs is therefore made on test scores. This covers the peak of each family, the best classifier by median, the best configurations (Table A2), the best model per corpus (Section 4.4), the Pareto frontiers, the best and cheapest runs in Table 8, and the best runs with and without class embeddings (Section 4.6). These results are descriptive test-set maxima. As estimates of what a configuration chosen in advance would score on new data they are optimistic. The optimism grows with the number of configurations a maximum is taken over, which favours the classical family (171 - 244 configurations per classifier and corpus) over the neural ones (4 - 25). The results show which configurations reached which scores on these test sets. Used as prospective guidance for choosing a classifier, they identify candidates to validate on the new task, not scores to expect from it.

Statistical tests. With eight corpora, we compare families with the exact two-sided Wilcoxon signed-rank test on per-corpus peak differences [24] [81]. We compare the eight classifiers with the Friedman test [82] followed by the Nemenyi post-hoc test on median scores [24]. For the class-embedding analyses we pair the best run with and the best run without the embeddings that share a corpus, classifier and language model (and, for static networks, the frozen-or-tuned setting). Pairs from the same corpus share its test set and are not independent observations, so we do not test the pairs directly. Instead, we average the paired differences within each corpus and apply the exact two-sided Wilcoxon test to the eight per-corpus means, as for the family comparisons. The pair counts and median pair differences are reported as descriptive statistics. With per-corpus medians instead of means, every conclusion at the 5% level is unchanged except one: for *solo* on Macro-F1, p falls from 0.055 to 0.039. In one comparison the two arms differ in size: for *solo*, the arm without class embeddings has one run per pair and the arm with them two or three, which favours the latter. Following the test-selection guidance of [83], we use non-parametric tests, because normality of the per-corpus differences cannot be assumed with eight corpora, and we report exact p-values. Each configuration was run once, so differences within a single configuration cannot be tested; variation across configurations is reported instead.

Frontiers and tolerance. For every corpus, the Pareto frontier is the set of runs that no other run beats on both score and training time. We also report the cheapest run within a tolerance of 1, 3 and 5 Macro-F1 points of the best, and its training time as a share of the best run's.

4. Results

4.1. Peak Effectiveness by Family

Table 5 gives each family's peak Macro-F1 on each corpus, and Figure 3 plots how far the best classical and static-embedding runs fall behind the best transformer. Three patterns are clear.

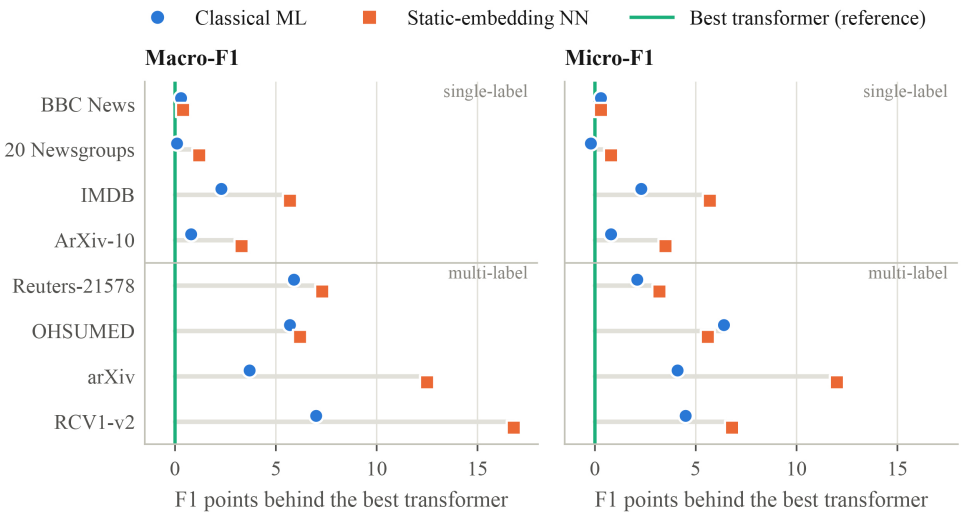

Figure 3. Distance of the best classical (circles) and best static-embedding (squares) run from the best fine-tuned transformer run on each corpus, in F1 points. Single-label corpora are above the rule, multi-label corpora below. Values are in Table 5 and Table A1.

Table 5. Peak Macro-F1 by family on each corpus: score of the best run (selected on test score; Section 3.6), the classifier (for transformers, model and head) that produced it, and its training time in seconds. The last two columns give the transformer's lead in F1 points. Micro-F1 results are in Table A1.

Corpus	Classical ML			Static-embedding NN			Transformer			Lead (pts)	
	F1	clf	time	F1	clf	time	F1	model/head	time	vs ML	vs NN
BBC News	0.989	LR	1.72	0.988	CNN	7.46	0.992	RoBERTa/CNN	141	+0.3	+0.4
20 Newsgroups	0.705	SVM	64.8	0.694	CNN	111	0.706	DeepSeek/SC	15,469	+0.1	+1.2
IMDB	0.925	LR	198	0.891	ATTN	88.1	0.948	RoBERTa/SC	2366	+2.3	+5.7
ArXiv-10	0.857	LR	2569	0.832	ATTN	403	0.865	DeepSeek/SC	42,048	+0.8	+3.3
Reuters-21578	0.606	SVM	16.7	0.592	ATTN	195	0.665	XLNet/SC	6095	+5.9	+7.3
OHSUMED	0.691	SVM	13.2	0.686	CNN	194	0.748	DeepSeek/CNN	28,201	+5.7	+6.2
arXiv	0.784	LR	466	0.696	LSTM	89.2	0.821	DeepSeek/CNN	12,254	+3.7	+12.5
RCV1-v2	0.648	LR	11,127	0.550	ATTN	711	0.718	RoBERTa/CNN	30,962	+7.0	+16.8
Mean	0.776			0.741			0.808			+3.2	+6.7

Table 6. Family comparisons across the eight corpora. Peak lead: mean difference in peak F1 (points), with range. W/T/L: corpora won, tied and lost by the first family. *p*: exact two-sided Wilcoxon signed-rank test on the per-corpus peak differences (ties dropped). Typical lead: mean difference between the families' typical (median) scores.

	Peak lead	Range	W/T/L	*p*	Typical lead
*Macro-F*1					
Transformer vs. classical ML	+3.2	[+0.1, +7.0]	8/0/0	0.0078	+7.8
Transformer vs. static NN	+6.7	[+0.4, +16.8]	8/0/0	0.0078	+6.5
Classical ML vs. static NN	+3.5	[+0.1, +9.8]	8/0/0	0.0078	−1.2

Continued

*Micro-F*1					
Transformer vs. classical ML	+2.5	[−0.2, +6.4]	7/0/1	0.0156	+6.5
Transformer vs. static NN	+4.7	[+0.3, +12.0]	8/0/0	0.0078	+4.8
Classical ML vs. static NN	+2.2	[−0.8, +7.9]	6/1/1	0.0312	−1.7

First, fine-tuned transformers have the highest peak Macro-F1 on all eight corpora. Their mean peak is 0.808, against 0.776 for the classical family and 0.741 for static-embedding networks. The advantage over each of the other families is consistent in sign ($p = 0.0078$, the smallest attainable p with eight corpora; Table 6) but modest in size: 3.2 points over the classical family and 6.7 points over the static networks on average. For Micro-F1 the leads shrink to 2.5 and 4.7 points (Table A1), and on 20 Newsgroups the best classical run has the higher Micro-F1. The transformer peak comes from DeepSeek-R1-Distill on four corpora, from RoBERTa on three and from XLNet on one; Section 4.4 asks what the larger models cost.

Second, the best classical run beats the best static-embedding run on all eight corpora for Macro-F1 (mean +3.5 points, $p = 0.0078$) and wins six of eight for Micro-F1, with one tie and one loss. This reverses the ordering usually assumed between linear models and neural networks over word vectors. It depends, however, on which features the linear model receives. The winning classical configuration on six corpora concatenates sparse features with mean-pooled embeddings from a frozen transformer, most often Llama-3.2-1B (Table A2). Only on OHSUMED (TF-IDF alone) and Reuters-21578 (TF-IDF with pooled fastText vectors) does the best classical configuration use no transformer at all.

Third, the transformer premium depends on the task type. On the four single-label corpora the best classical run trails the best transformer by 0.1 - 2.3 Macro-F1 points (mean 0.9). On the four multi-label corpora, which have 23 - 115 classes and long-tailed label distributions, it trails by 3.7 - 7.0 points (mean 5.6). With four corpora per group the separation is suggestive rather than conclusive (exact Mann-Whitney $p = 0.029$ for Macro-F1 and 0.057 for Micro-F1). Multi-label status is also confounded with the number of classes and with corpus size. The pattern is nevertheless consistent with Macro-F1 rewarding performance on rare classes, where a fine-tuned encoder's contextual features help most.

4.2. Typical Configurations and Robustness

Peak scores reward exhaustive search, and the classical module searches far more broadly than the neural ones. Typical scores reverse part of the picture (Table 6 last column). The median configuration of the best transformer classifier leads the median classical configuration by 7.8 Macro-F1 points, more than twice the peak lead, while the median static-embedding network edges out the median classical configuration by 1.2 points. In other words, classical models reach transformer-like scores only when their features are chosen well; many of their configurations, including all naive Bayes runs, are much weaker. Transformers are comparatively insensitive to the choice of model and head.

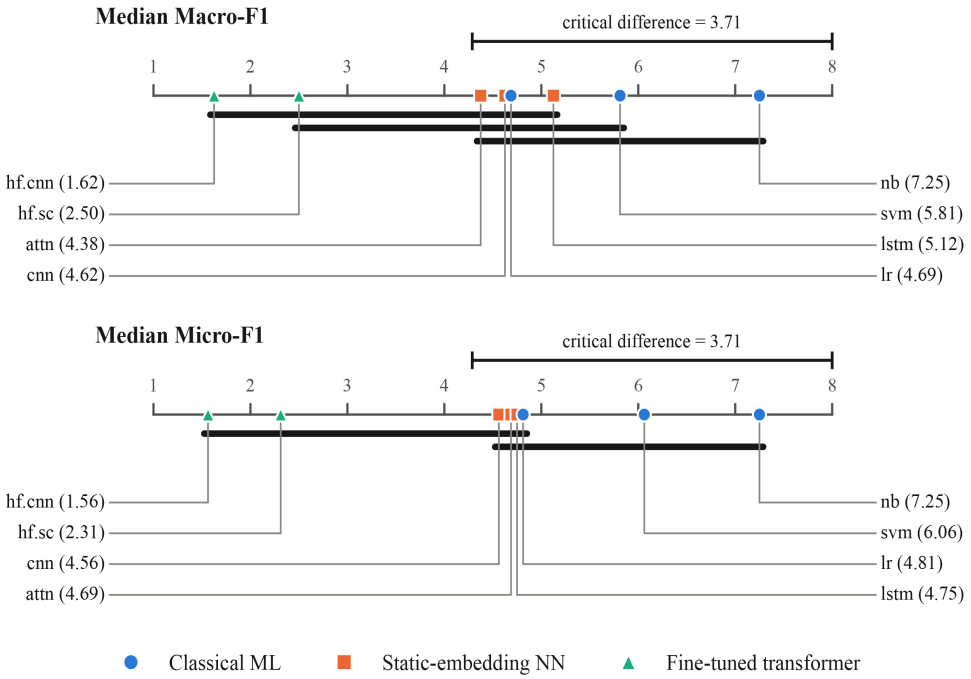

Figure 4. Critical-difference diagrams: average rank of each classifier by median score across the eight corpora (1 = best). Classifiers joined by a bar are not significantly different (Nemenyi, α = 0.05, critical difference 3.71). Marker shape and colour give the family.

Figure 4 ranks all eight classifiers by median score on each corpus. The Friedman test rejects equal performance for both Macro-F1 ($\chi_7^2 = 29.9$, $p = 1.0\times10^{-4}$) and Micro-F1 ($\chi_7^2 = 31.9$, $p = 4.3\times10^{-5}$). With eight corpora, however, the Nemenyi critical difference is 3.71 ranks, so only the largest gaps are resolved:

- Macro-F1: HF-CNN ranks significantly above the linear SVM and NB, and HF-SC above NB.
- Micro-F1: both transformer heads rank above the SVM and NB.

No difference between the transformer heads and the static networks, or between the static networks and LR, is significant at this sample size.

Medians also guard against a failure mode visible in the run log. Of the 528 neural runs, nine have a Macro-F1 below 0.35:

- On arXiv, three HF-SC runs (BERT, and DistilBERT with and without TCEs) score 0.000, and five LSTM runs score between 0.000 and 0.348.
- On IMDB, one Llama-3.2-1B HF-CNN run predicts a single class: Macro-F1 0.333, the score of a constant classifier on a balanced binary task.

These runs pull the HF-SC mean on arXiv to 0.547 against a median of 0.681. Averaged results, as reported in [23], are sensitive to such runs; peak and median results are not.

4.3. What the Last Points Cost

Table 7 sets the transformer's lead on each corpus against the typical training time of each family. A typical transformer run trains 25 - 45 times longer than a typical static-embedding run (geometric mean 32×). Counting the four GPUs it held, that

is 127 times the accelerator-time. It trains 24 - 133 times longer than a typical classical run (geometric mean 75×). The winning runs are costlier still, because the transformer peak is often reached by one of the two large models: in wall-clock time, the best transformer run took 19 - 146 times as long as the best static-embedding run (geometric mean 61×).

Table 7. The transformer's peak Macro-F1 lead (points) and the median training time (seconds) of each family's runs on each corpus, with the ratio of the transformer median to the static-embedding and classical medians. Multiply the first ratio by four for accelerator-time.

Corpus	Lead (pts)		Median training time (s)			Transformer time	
	vs ML	vs NN	ML	NN	Transf.	/NN	/ML
BBC News	+0.3	+0.4	2.79	5.83	222	38×	80×
20 Newsgroups	+0.1	+1.2	26.9	60.5	2204	36×	82×
IMDB	+2.3	+5.7	34.7	119	3018	25×	87×
ArXiv-10	+0.8	+3.3	141	324	9106	28×	64×
Reuters-21578	+5.9	+7.3	34.9	182	4624	25×	133×
OHSUMED	+5.7	+6.2	75.9	182	5356	30×	71×
arXiv	+3.7	+12.5	34.2	130	3999	31×	117×
RCV1-v2	+7.0	+16.8	1363	723	32,649	45×	24×
Geometric mean						32×	75×

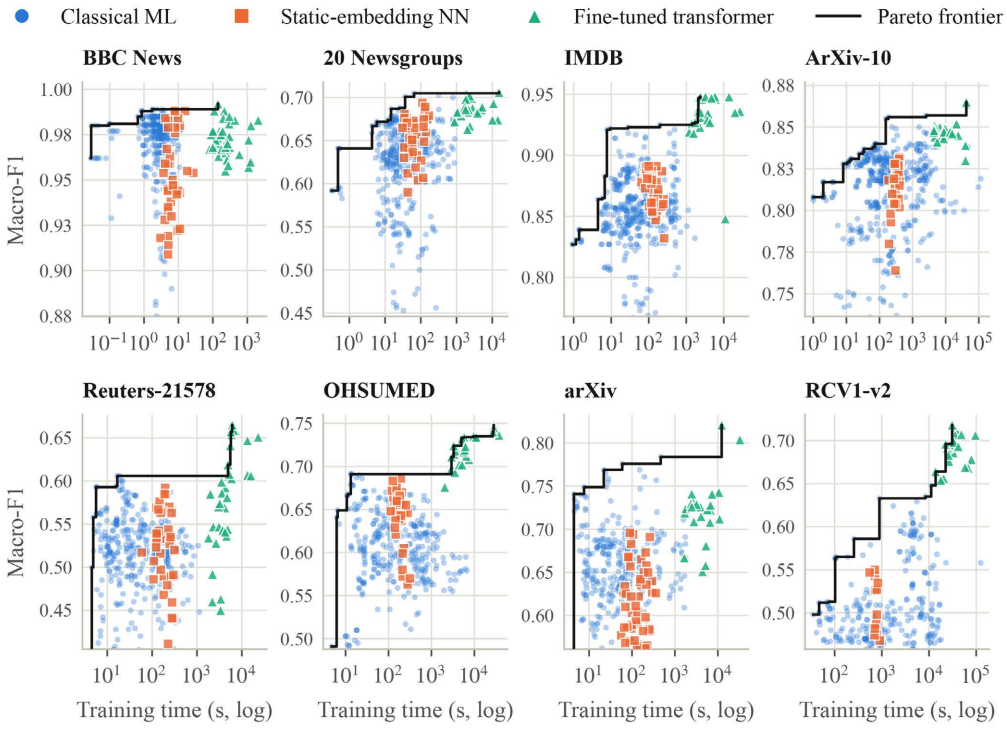

Figure 5. Macro-F1 against training time (log scale) for every run on each corpus; the line is the Pareto frontier. Classical runs are drawn small and translucent because there are several hundred per corpus. Vertical axes differ between panels and are cropped to the upper part of each corpus's range.

Ratios between families hide the runs in between. Figure 5 plots every run against its training time and draws the Pareto frontier. On every corpus the frontier ends at a transformer run, so a transformer is always the most accurate choice. Below that point, classical runs dominate. Across the eight corpora, 96 runs lie on a Macro-F1 frontier: 71 classical and 25 transformer. No static-embedding run does; for Micro-F1, four do, on OHSUMED and RCV1-v2. The frontiers have two shapes:

- On the single-label corpora they flatten early: the best classical run comes within 0.1 - 2.3 points of the best transformer run at 12 to 239 times less training time.
- On Reuters-21578, OHSUMED, arXiv and RCV1-v2 they end in a tall final step, where transformer runs add several points at the far right of the time axis.

Table 8 reads the frontier at fixed tolerances: for a given tolerance, what is the cheapest run and what share of the best run's classifier-fitting time does it need?

- **Single-label corpora.** A tolerance of one point is enough to replace the transformer on BBC News, 20 Newsgroups and ArXiv-10 with a linear model that needs 0.23% - 0.47% of the fitting time. On IMDB a three-point tolerance does the same at 0.32%.
- **Multi-label corpora.** No tolerance up to three points admits anything but a transformer. However, a base-size encoder often comes close to the large model that holds the peak: on OHSUMED, RoBERTa comes within 2.4 points at 12% of DeepSeek's training time. At five points a classical SVM appears on arXiv at 0.48% of the time, but on Reuters-21578, OHSUMED and RCV1-v2 even a five-point tolerance still requires a transformer.

Table 8. Cheapest run within 1, 3 and 5 Macro-F1 points of the best run on each corpus: model, its Macro-F1, and its training (classifier-fitting) time as a share of the best run's (—: only the best run qualifies). Runs are selected on test scores (Section 3.6). Classical entries give the classifier and the language model whose embeddings, projections or tokenizer the features use; neither feature extraction nor inference is timed for any run.

Corpus	Best run	Time (s)	Within 1 pt	Within 3 pts	Within 5 pts
BBC News	RoBERTa/CNN 0.992	141	SVM, word2vec 0.985 (0.47%)	NB, TF-IDF 0.962 (0.02%)	NB, TF-IDF 0.962 (0.02%)
20 Newsgroups	DeepSeek/SC 0.706	15,469	LR, Llama 0.701 (0.23%)	LR, RoBERTa 0.678 (0.09%)	SVM, TF-IDF 0.659 (0.03%)
IMDB	RoBERTa/SC 0.948	2366	RoBERTa/CNN 0.947 (89%)	LR, RoBERTa 0.921 (0.32%)	LR, RoBERTa 0.921 (0.32%)
ArXiv-10	DeepSeek/SC 0.865	42,048	LR, DeepSeek 0.855 (0.37%)	LR, fastText 0.835 (0.07%)	NB, TF-IDF 0.817 (<0.01%)
Reuters-21578	XLNet/SC 0.665	6095	XLNet/SC 0.657 (93%)	XLNet/CNN 0.641 (91%)	RoBERTa/CNN 0.619 (81%)
OHSUMED	DeepSeek/CNN 0.748	28,201	Llama/SC 0.740 (96%)	RoBERTa/CNN 0.724 (12%)	DistilBERT/CNN 0.711 (10%)
arXiv	DeepSeek/CNN 0.821	12,254	—	—	SVM, Llama 0.776 (0.48%)
RCV1-v2	RoBERTa/CNN 0.718	30,962	—	RoBERTa/SC 0.696 (73%)	RoBERTa/SC 0.696 (73%)

The savings in this section are savings in classifier-fitting time, not in deployment cost, and two costs that bear on deployment are left out. First, the classical runs that use transformer features depend on a forward pass of the frozen transformer over the training and test documents, which is not timed. That pass is cheaper than fine-tuning, since it involves no backward pass and a single epoch, but it is not free; on RCV1-v2, whose test set has 781,265 documents, it is substantial. Second, inference on new documents is excluded for every family, and leaving it out does not make the comparisons conservative. The cheaper runs that Table 8 identifies on 20 Newsgroups, IMDB and ArXiv-10 use frozen transformer embeddings. On IMDB and ArXiv-10 they come from the same model as the best fine-tuned run (RoBERTa and DeepSeek-R1-Distill). On 20 Newsgroups they come from Llama-3.2-1B, close in size to the DeepSeek-R1-Distill model it replaces. In deployment these configurations need a transformer forward pass for every new document, as the fine-tuned model does, so their deployment savings remain unmeasured. Only the classical entries over TF-IDF, LSA or static word vectors avoid transformer inference, for example the run within one point on BBC News (a linear SVM over word2vec features); a transformer tokenizer, where one is used, adds no forward pass.

4.4. Which Transformer?

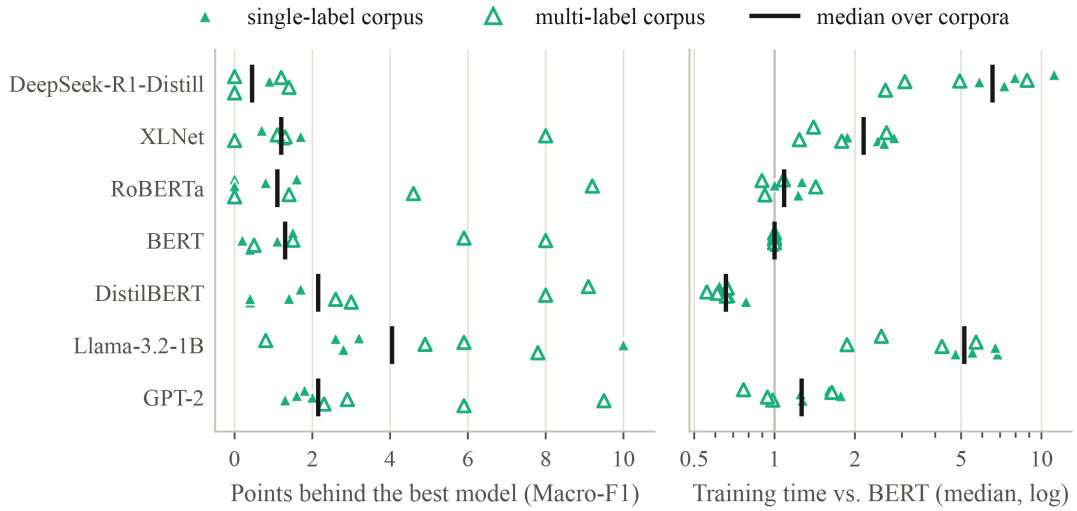

Figure 6. Fine-tuned transformer models: Macro-F1 points behind the best model on each corpus (left) and median training time relative to BERT (right, log scale). Each marker is one corpus; bars are medians. Filled markers are single-label corpora, open markers multi-label.

Within the transformer family, model size and training time vary by an order of magnitude. Table 9 and Figure 6 compare the seven models by their best Macro-F1 on each corpus and by their median training time relative to BERT on the same corpus. The models differ ($\chi_6^2 = 18.9$, $p = 0.004$, Friedman), but not in proportion to their cost:

- **DeepSeek-R1-Distill** has the best average rank (2.38) and wins four corpora,

but takes a median 6.6 times as long as BERT.

- **RoBERTa** wins three corpora at 1.1 times BERT's time, and **XLNet** two at 2.2 times (the two tie on IMDB).
- **Llama-3.2-1B** ranks sixth of seven, on average 4.7 points behind the best model, at 5.1 times BERT's time.

Table 9. Fine-tuned transformer models across the eight corpora (Macro-F1; best run per model and corpus over both heads, with and without TCEs). Rank: average rank (1 = best). Wins: corpora on which the model is best. Behind: mean and maximum points behind the best model on the corpus. Time: median training time relative to BERT on the same corpus.

Model	Rank	Wins	Behind (mean)	Behind (max)	Time
DeepSeek	2.38	4	0.6	1.4	6.56×
XLNet	2.88	2	1.8	8.0	2.16×
RoBERTa	3.25	3	2.2	9.2	1.09×
BERT	3.38	0	2.4	8.0	1.00×
DistilBERT	4.94	0	3.3	9.1	0.66×
Llama	5.50	0	4.7	10.0	5.14×
GPT-2	5.69	0	3.4	9.5	1.26×

Comparing, on each corpus, the better of the two large decoders with the best base-size encoder (BERT, RoBERTa, DistilBERT or XLNet):

- the large models gain 8.0 points on arXiv, 1.3 on ArXiv-10 and OHSUMED, and 0.2 on 20 Newsgroups;
- they lose 0.9 - 1.4 points on the other four corpora (median change −0.4, mean +0.8);
- they need 2.3 - 7.2 times the training time (geometric mean 4.9).

The one clear win, arXiv, is a corpus of long, technical abstracts on which DeepSeek-R1-Distill, whose base model was trained for mathematical reasoning, leads every other model by at least 7.8 points. Elsewhere, the extra parameters buy little. DistilBERT trains in two-thirds of BERT's time and trails the best base encoder by a median of 1.4 points, though by 8.0 on Reuters-21578, the corpus with the most classes.

4.5. Representation Choices for Classical Models

Within the classical family, which representation pays? Figure 7 compares pooled embeddings alone (*solo*, 300 - 2048 dimensions) with the best representation form on each corpus. On six corpora, pooled embeddings come within 0.1 - 2.1 Macro-F1 points of the best form, usually *cat-doc* (sparse features concatenated with pooled embeddings). On ArXiv-10, IMDB and RCV1-v2 the extra 0.1 - 1.5 points cost 8 - 14 times the median training time. The exceptions are OHSUMED, where

the best form is TF-IDF alone, 7.5 points ahead of pooled embeddings and faster, and Reuters-21578, where *cat-doc* leads by 3.9 points.

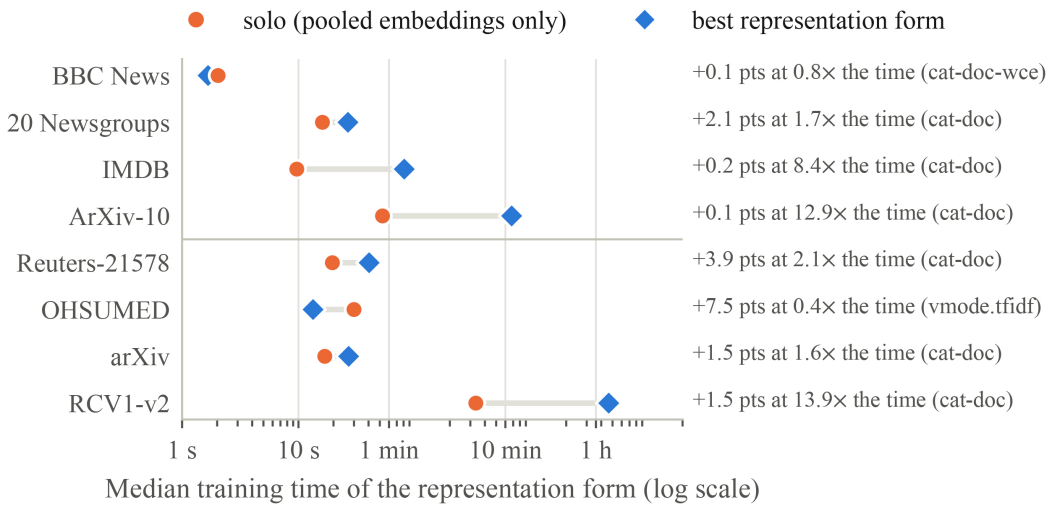

Figure 7. Pooled embeddings alone (*solo*) against the best classical representation form on each corpus (Macro-F1 of the best run of each form; median training time of each form's runs).

4.6. Class Embeddings

Table 10 and Figure 8 summarise every matched comparison of a model with and without class embeddings. Pair counts and medians describe the individual pairs. The *p*-values come from the exact Wilcoxon test on the eight per-corpus mean differences (Section 3.6), so the smallest attainable value is 0.0078.

Table 10. Effect of class embeddings on matched runs (best run with vs. without, same corpus, classifier and language model). Median change in F1 points over pairs, number of pairs and of corpora (by mean change) that improved/worsened, exact two-sided Wilcoxon *p* over the eight per-corpus mean changes, and median training-time multiple. Classical rows give the base form and what its *-wce* variant adds: the WCE columns alone for *dot*, and the projection $\Pi_{M+\mathrm{WCE}}$ (projected embedding features and WCEs together) for the others.

	Macro-F1					Micro-F1				
			Up/down				Up/down			
Setting	Pairs	Median	pairs	corpora	*p*	Median	pairs	corpora	*p*	Time
WCE, static-embedding NNs	129	+0.8	80/45	5/3	0.31	+0.4	72/51	4/4	0.55	1.0×
Classical, dot + WCE	158	+10.9	153/3	8/0	0.008	+11.1	156/0	8/0	0.008	1.0×
Classical, solo + $\Pi_{M+\mathrm{WCE}}$	158	+3.5	118/37	6/2	0.055	+4.1	126/28	7/1	0.023	2.6×
Classical, lsa + $\Pi_{M+\mathrm{WCE}}$	158	+0.1	80/75	5/3	0.46	+0.8	90/63	5/3	0.20	2.6×
Classical, cat-doc + $\Pi_{M+\mathrm{WCE}}$	149	−2.1	41/102	1/7	0.016	−1.9	36/103	1/7	0.016	2.1×
TCE, fine-tuned transformers	79	0.0	39/35	5/3	0.55	0.0	39/31	5/3	0.38	1.0×
excluding collapsed runs	77	0.0	38/35	4/4	0.84	0.0	38/31	4/4	0.95	1.0×

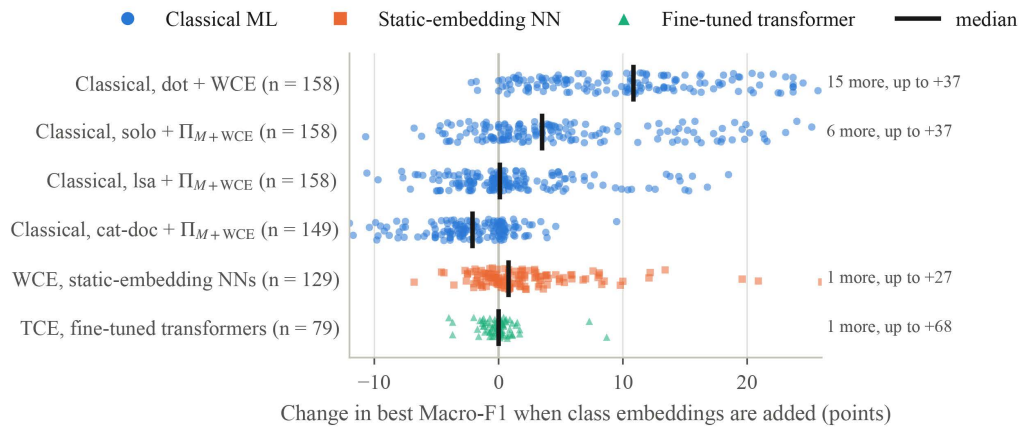

Figure 8. Change in best Macro-F1 when class embeddings are added, for every matched pair; for the classical *solo*, *lsa* and *cat-doc* rows, what is added is the projection $\Pi_{M+\mathrm{WCE}}$, *i.e.* projected embedding features together with WCEs. Bars are medians. Pairs beyond the right edge are counted in the margin; the single TCE outlier (+68) is a pair in which the run without TCEs collapsed.

Static-embedding networks. Across 129 pairs (same corpus, network, word model and frozen-or-tuned setting), WCEs raise Macro-F1 by a median of 0.8 points (mean 2.3) and improve 80 pairs against 45, at no extra training time. The gains are uneven across corpora. The mean change is largest on arXiv (+8.6) and Reuters-21578 (+4.8), the corpora with the most classes outside RCV1-v2. It is smaller on 20 Newsgroups (+2.4), OHSUMED (+1.4) and BBC News (+0.1), and slightly negative on ArXiv-10, IMDB and RCV1-v2. With five corpora up and three down, the effect is not significant across corpora ($p = 0.31$; 0.55 for Micro-F1). Treating the 129 pairs as independent would give $p = 1.5\times10^{-6}$. That figure overstates the evidence for an effect that generalises across tasks, because pairs from the same corpus share its test set. The positive pair-level median is consistent with the gains reported by [25]. In these runs, however, the gain does not hold across corpora.

Classical models. Only one of the four comparisons isolates the WCEs (Section 3.3). *dot-wce* differs from *dot* by the WCE columns $\boldsymbol{x}_d W$ alone. Adding them raises Macro-F1 in 153 of 158 pairs, by a median of 10.9 points, and on all eight corpora ($p = 0.008$). The other three *-wce* forms append the whole projection $\Pi_{M+\mathrm{WCE}}(d)$ to their base form. They thus add the unsupervised projected embedding features $\boldsymbol{x}_d E_M$ and the WCEs together, so their differences cannot be attributed to the WCEs alone:

- appending the projection to pooled embeddings (*solo*) raises Macro-F1 by a median of 3.5 points in 118 of 158 pairs, and on six corpora against two ($p = 0.055$; for Micro-F1 seven against one, $p = 0.023$). This comparison also favours the *-wce* arm, which has more runs (Section 3.6);
- appending it to LSA features changes Macro-F1 by a median of +0.1 points, with five corpora up and three down ($p = 0.46$), so no statistically detectable systematic change was found;
- appending it to *cat-doc* lowers Macro-F1 in 102 of 149 pairs (median −2.1) and

on seven of eight corpora ($p = 0.016$).

These variants also take up to 2.6 times the training time. The WCE columns thus strengthen the weakest representation, the plain projection. For the other three forms, these runs cannot separate the contribution of the WCEs from that of the projected embedding features or of the larger feature vector.

Fine-tuned transformers. For TCEs, no statistically detectable systematic effect was found in these runs. Across 79 pairs (same corpus, model, vocabulary and head), the median change in Macro-F1 is 0.0 points: 39 pairs improve, 35 worsen and 5 tie. The per-corpus mean change is positive on five corpora and negative on three ($p = 0.55$; 0.38 for Micro-F1). The pair-level mean (+0.9) is carried by one arXiv pair in which the run without TCEs collapsed. Excluding the two pairs with a collapsed run, the mean change is 0.00 points ($p = 0.84$). A nonsignificant test does not establish that runs with and without TCEs perform equivalently. With eight corpora the test has little power. Outside arXiv, the per-corpus mean changes range from −1.1 (RCV1-v2) to +0.7 points (Reuters-21578), so effects of about a point in either direction cannot be ruled out. The runs do not, however, support the earlier report that TCEs consistently *degraded* transformer performance [23]. One possible reason why no effect was detected is that fine-tuning already shapes the encoder's representation around the label set, so token-class statistics fused after the encoder may add little that the head does not already see. Injecting class information earlier, for instance into intermediate layers through a learned projection, remains untested.

5. Discussion

5.1. Choosing a Classifier

The results suggest a procedure for choosing a classifier, with two qualifications. First, its cost claims concern classifier-fitting time only (Section 4.3). Feature extraction and inference were not timed, and the frozen-transformer configurations it starts from need transformer inference on every new document, so their deployment savings are unmeasured. Second, the configurations and scores it rests on are test-set maxima (Section 3.6). It therefore indicates which configurations to try and validate on a new task, not the scores they will reach there.

1) Start with a linear model over TF-IDF features concatenated with mean-pooled embeddings from a frozen transformer; logistic regression and a linear SVM are both strong. On the single-label corpora here, the best such linear configurations came within one to three points of the best fine-tuned transformer at well under one percent of its fitting time.

2) For a multi-label task with many classes, especially when Macro-F1 (performance on rare classes) matters, fine-tuning added four to seven points here.

3) If you fine-tune, start with a base-size encoder such as RoBERTa or BERT. Here the billion-parameter-scale decoders cost about five times as much to fine-tune for no consistent gain; the exception was the technical arXiv corpus.

4) These runs give no evidence that token-class embeddings, fused after the en-

coder, help a fine-tuned transformer. In static-embedding networks, WCEs helped on the many-class arXiv and Reuters-21578 corpora but not consistently elsewhere. In linear models, WCE columns strengthened the weak *dot* projection on every corpus.

Static-embedding neural networks occupy an awkward middle: on these corpora they were never more accurate than the best linear configuration and never on the Macro-F1 frontier of score against fitting time.

A classifier that is retrained whenever labels drift or categories change pays its fitting cost repeatedly, so a typical fitting-time ratio of 30 to 75 compounds over the life of the system. Inference cost also recurs with every document classified, and it is not measured here. For the frozen-transformer configurations it includes a transformer forward pass per document, so their advantage in deployment may be considerably smaller than the fitting-time ratio. Where the cheaper model is within the tolerance the application can bear, Table 8 identifies a candidate to validate and to cost in deployment.

5.2. Why Frozen Embeddings and Linear Models Do So Well

Contextual token representations are highly anisotropic [84], which carries over to their mean-pooled averages. Sentence-embedding work also shows that naively pooling a model trained for a different objective gives weak sentence vectors [85]. Yet for topic and sentiment classes that are separable by vocabulary and broad semantics, a linear model over such features, alongside the sparse TF-IDF vector that keeps exact lexical evidence, is enough. Pooled features and sparse features are complementary: the best classical configuration on six corpora uses both. What fine-tuning adds is a representation shaped by the label set. That matters most when there are many classes, some of them rare and distinguished by subtle contextual cues, which is the multi-label regime in our data.

5.3. Limitations

- **Heterogeneous hardware.** The families ran on different machines and the transformers on three GPU types (Table 4). The ratios are therefore costs on this equipment, not hardware-normalised efficiencies.
- **What is timed.** Training time excludes feature extraction, including the frozen transformer pass that the best classical configurations depend on, and excludes inference for all families. The reported savings are therefore classifier-fitting savings. Deployment savings are not measured, and for the frozen-transformer configurations, which need transformer inference on new documents, they may be much smaller.
- **Test-set selection.** The logs contain no validation scores, so every reported winner is a test-set maximum (Section 3.6). Peak scores and tolerance choices are optimistic as estimates for a configuration chosen in advance, most of all for the classical family, which has the largest search.
- **Unequal search budgets.** The classical module explores 171 - 244 configura-

tions per corpus for each of LR and the SVM, the neural modules 4 - 25 per classifier. This favours the classical family at peak and penalises it in typical comparisons, which is why we report both.

- **Single runs.** Each configuration was run once. Peak scores are therefore maxima over configurations, not means over seeds, and on the closest comparisons (BBC News, 20 Newsgroups) differences of a few tenths of a point are within the range that seed variation could produce.
- **Defaults and scale.** Classical models used default hyperparameters; tuning them would likely help them further. The transformers are at most 1.5 B parameters, fine-tuned in full with DataParallel. Parameter-efficient fine-tuning, distributed data parallelism and larger models would change both sides of the trade-off.
- **Time precision.** For 11% of runs the training time is available only rounded to the second, which matters only for the fastest classical runs.

6. Conclusions

How much accuracy does fine-tuning a transformer buy for text classification, and at what cost? Across 3725 runs on eight corpora, the answer is: on every corpus some, but on average little relative to a well-chosen linear model (3.2 Macro-F1 points), at a typical training-time multiple of 75. The average hides a clear split. On single-label tasks, a linear model over frozen, pooled transformer features came within 0.5 - 2.7 points at well under one percent of the best run's classifier-fitting time. Because those features also require transformer inference on new documents, the deployment savings remain to be measured. On multi-label, many-class tasks, the transformer's lead of four to seven points has no linear substitute, though a base-size encoder is usually enough. Static-embedding networks were dominated throughout, and billion-parameter-scale decoders rarely repaid their cost. Word-class embeddings in static networks helped on some corpora but not consistently across them. For token-class embeddings in fine-tuned transformers, no statistically detectable systematic effect was found in these runs. All of these comparisons are between test-set maxima. These differences become visible only by reporting peak and typical scores, paired tests, accelerator-time, Pareto frontiers and tolerance tables, rather than averages over unequal numbers of configurations. We recommend the same protocol for other comparisons of model families.

Three extensions would sharpen these conclusions. The first is to time feature extraction and inference, so that the frozen-feature configurations and deployed transformers are costed in full. The second is to select configurations on held-out validation data and to repeat runs over seeds, so that reported winners become estimates of prospective performance rather than test-set maxima. The third is to add the efficient alternatives that have matured since these runs: parameter-efficient fine-tuning [64] [65], modern efficient encoders such as ModernBERT [86], few-shot sentence-embedding classifiers such as SetFit [87], and prompted LLMs

[22]. Layer Cake and the analysis scripts released with this article are designed to make each of these a configuration change rather than a new study.

Data and Code Availability

The Layer Cake framework is available at https://github.com/pworth1971/layer-cake. The run-level results of the original study, the summary tables of the earlier paper, and the scripts that produce every table and figure in this article (extract_tables.py, extract_runs.py, analyze.py, tables.py, figures.py) are in the same repository.

Acknowledgements

In preparing this revision, the author used Claude Code (Anthropic), an AI assistant for software and writing tasks, through its Visual Studio Code extension, with the Claude Opus 5.5 model (claude-opus-5-5), on 23 and 29 October 2026 and, for the proof corrections, the Claude Fable 5.2 model (claude-fable-5-2) on 22 November 2026. The assistant was used to:

- redraft and restructure the manuscript from the earlier version [23], and revise the analysis and text in response to editorial review;
- write the analysis scripts released with this article, re-analyse the published run-level results with them, and check that they reproduce every summary statistic reported in [23];
- check each reference against Crossref, arXiv, the ACL Anthology and publisher records, and correct the bibliography;
- regenerate the figures and tables, and typeset the manuscript in the journal template.

No new experiments were run, and the assistant neither generated nor altered any experimental result. The author reviewed and edited all AI-assisted material and takes full responsibility for the content of this article.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Devlin, J., Chang, M., Lee, K. and Toutanova, K. (2019) BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. *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), Minneapolis, 2 June-7 June 2019, 4171-4186. https://doi.org/10.18653/v1/n19-1423

[2] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I. (2017) Attention Is All You Need. *Advances in Neural Information Processing Systems*, **30**, 5998-6008. https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[3] Howard, J. and Ruder, S. (2018) Universal Language Model Fine-Tuning for Text

Classification. *Proceedings of the* 56*th Annual Meeting of the Association for Computational Linguistics* (Volume 1: Long Papers), Melbourne, 15-20 July 2018, 328-339. https://doi.org/10.18653/v1/p18-1031

[4] Sun, C., Qiu, X., Xu, Y. and Huang, X. (2019) How to Fine-Tune BERT for Text Classification? In: Sun, M., Huang, X., Ji, H., Liu, Z. and Liu, Y., Eds., *Lecture Notes in Computer Science*, Springer International Publishing, 194-206. https://doi.org/10.1007/978-3-030-32381-3_16

[5] Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., *et al.* (2024) The Llama 3 Herd of Models. arXiv:2407.21783.

[6] Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., *et al.* (2025) DeepSeek-R1 Incentivizes Reasoning in LLMs through Reinforcement Learning. *Nature*, **645**, 633-638. https://doi.org/10.1038/s41586-025-09422-z

[7] Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M. and Gao, J. (2021) Deep Learning—Based Text Classification: A Comprehensive Review. *ACM Computing Surveys*, **54**, 1-40. https://doi.org/10.1145/3439726

[8] Li, Q., Peng, H., Li, J., Xia, C., Yang, R., Sun, L., *et al.* (2022) A Survey on Text Classification: From Traditional to Deep Learning. *ACM Transactions on Intelligent Systems and Technology*, **13**, 1-41. https://doi.org/10.1145/3495162

[9] Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L. and Brown, D. (2019) Text Classification Algorithms: A Survey. *Information*, **10**, Article 150. https://doi.org/10.3390/info10040150

[10] Strubell, E., Ganesh, A. and McCallum, A. (2019) Energy and Policy Considerations for Deep Learning in NLP. *Proceedings of the* 57*th Annual Meeting of the Association for Computational Linguistics*, Florence, 28 July-2 August 2019, 3645-3650. https://doi.org/10.18653/v1/p19-1355

[11] Schwartz, R., Dodge, J., Smith, N.A. and Etzioni, O. (2020) Green AI. *Communications of the ACM*, **63**, 54-63. https://doi.org/10.1145/3381831

[12] Dodge, J., Gururangan, S., Card, D., Schwartz, R. and Smith, N.A. (2019) Show Your Work: Improved Reporting of Experimental Results. *Proceedings of the* 2019 *Conference on Empirical Methods in Natural Language Processing and the* 9*th International Joint Conference on Natural Language Processing* (EMNLP-IJCNLP), Hong Kong SAR, 3-7 November 2019, 2185-2194. https://doi.org/10.18653/v1/d19-1224

[13] Henderson, P., Hu, J., Romoff, J., Brunskill, E., Jurafsky, D. and Pineau, J. (2020) Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning. *Journal of Machine Learning Research*, **21**, 1-43.

[14] Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L.M., Rothchild, D., So, D., Texier, M. and Dean, J. (2021) Carbon Emissions and Large Neural Network Training. arXiv:2104.10350.

[15] Luccioni, S., Jernite, Y. and Strubell, E. (2024) Power Hungry Processing: Watts Driving the Cost of AI Deployment? *Proceedings of the* 2024 *ACM Conference on Fairness, Accountability, and Transparency* (FAccT '24), Rio de Janeiro, 3-6 June 2024, 85-99. https://doi.org/10.1145/3630106.3658542

[16] Joachims, T. (1998) Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In: Nédellec, C. and Rouveirol, C., Eds., *Lecture Notes in Computer Science*, Springer, 137-142. https://doi.org/10.1007/bfb0026683

[17] Wang, S. and Manning, C. (2012) Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. *Proceedings of the* 50*th Annual Meeting of the Association for Computational Linguistics* (Volume 2: Short Papers), Jeju Island, 8-14 July 2012, 90-94. https://aclanthology.org/P12-2018/

[18] Joulin, A., Grave, E., Bojanowski, P. and Mikolov, T. (2017) Bag of Tricks for Efficient Text Classification. *Proceedings of the* 15*th Conference of the European Chapter of the Association for Computational Linguistics*: *Volume* 2, *Short Papers*, Valencia, 3-7 April 2017, 427-431. https://doi.org/10.18653/v1/e17-2068

[19] Shen, D., Wang, G., Wang, W., Min, M.R., Su, Q., Zhang, Y., *et al.* (2018) Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms. *Proceedings of the* 56*th Annual Meeting of the Association for Computational Linguistics* (Volume 1: Long Papers), Melbourne, 15-20 July 2018, 440-450. https://doi.org/10.18653/v1/p18-1041

[20] Galke, L. and Scherp, A. (2022) Bag-of-Words vs. Graph vs. Sequence in Text Classification: Questioning the Necessity of Text-Graphs and the Surprising Strength of a Wide MLP. *Proceedings of the* 60*th Annual Meeting of the Association for Computational Linguistics* (Volume 1: Long Papers), Dublin, 22-27 May 2022, 4038-4051. https://doi.org/10.18653/v1/2022.acl-long.279

[21] Bucher, M.J.J. and Martini, M. (2024) Fine-Tuned 'Small' LLMs (Still) Significantly Outperform Zero-Shot Generative AI Models in Text Classification. arXiv:2406.08660.

[22] Edwards, A. and Camacho-Collados, J. (2024) Language Models for Text Classification: Is In-Context Learning Enough? *Proceedings of the Language Resources and Evaluation Conference*, Torino, 20-25 May 2024, 10058-10072. https://doi.org/10.63317/3dnziwdd5bh2

[23] Worth, Jr., P.J. and Cardei, I. (2025) Layer Cake: On Language Representation and Compute Characteristics in Text Classification. *International Journal of Intelligence Science*, **15**, 184-262. https://doi.org/10.4236/ijis.2025.154010

[24] Demšar, J. (2006) Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, **7**, 1-30.

[25] Moreo, A., Esuli, A. and Sebastiani, F. (2021) Word-Class Embeddings for Multiclass Text Classification. *Data Mining and Knowledge Discovery*, **35**, 911-963. https://doi.org/10.1007/s10618-020-00735-3

[26] Sparck Jones, K. (1972) A Statistical Interpretation of Term Specificity and Its Application in Retrieval. *Journal of Documentation*, **28**, 11-21. https://doi.org/10.1108/eb026526

[27] Salton, G. (1971) The SMART Retrieval System: Experiments in Automatic Document Processing. Prentice-Hall.

[28] Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K. and Harshman, R. (1990) Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, **41**, 391-407. https://doi.org/10.1002/(sici)1097-4571(199009)41:6<391::aid-asi1>3.0.co;2-9

[29] Bengio, Y., Ducharme, R., Vincent, P. and Jauvin, C. (2003) A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, **3**, 1137-1155.

[30] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. and Kuksa, P. (2011) Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, **12**, 2493-2537.

[31] Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013) Efficient Estimation of Word Representations in Vector Space. https://arxiv.org/abs/1301.3781

[32] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J. (2013) Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems*, **26**, 3111-3119.

[33] Pennington, J., Socher, R. and Manning, C. (2014) GloVe: Global Vectors for Word

Representation. *Proceedings of the* 2014 *Conference on Empirical Methods in Natural Language Processing* (EMNLP), Doha, 25-29 October 2014, 1532-1543. https://doi.org/10.3115/v1/d14-1162

[34] Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T. (2017) Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, **5**, 135-146. https://doi.org/10.1162/tacl_a_00051

[35] Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C. and Joulin, A. (2018) Advances in Pre-Training Distributed Word Representations. *Proceedings of the Language Resources and Evaluation Conference*, Miyazaki, 7-12 May 2018, 52-55. https://doi.org/10.63317/4b3prw5a5tze

[36] Harris, Z.S. (1954) Distributional Structure. *WORD*, **10**, 146-162. https://doi.org/10.1080/00437956.1954.11659520

[37] Firth, J.R. (1957) A Synopsis of Linguistic Theory, 1930-1955. *Studies in Linguistic Analysis* (*Special Volume of the Philological Society*), Blackwell, 1-32.

[38] Worth, P.J. (2023) Word Embeddings and Semantic Spaces in Natural Language Processing. *International Journal of Intelligence Science*, **13**, 1-21. https://doi.org/10.4236/ijis.2023.131001

[39] Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., *et al.* (2018) Deep Contextualized Word Representations. *Proceedings of the* 2018 *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, *Volume* 1 (Long Papers), New Orleans, 1-6 June 2018, 2227-2237. https://doi.org/10.18653/v1/n18-1202

[40] Bahdanau, D., Cho, K. and Bengio, Y. (2015) Neural Machine Translation by Jointly Learning to Align and Translate. https://arxiv.org/abs/1409.0473

[41] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V. (2019) RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv:1907.11692.

[42] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. and Le, Q.V. (2019) XLNet: Generalized Autoregressive Pretraining for Language Understanding. *Advances in Neural Information Processing Systems*, **32**, 5753-5763. https://proceedings.neurips.cc/paper_files/paper/2019/hash/dc6a7e655d7e5840e66733e9ee67cc69-Abstract.html

[43] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I. (2019) Language Models are Unsupervised Multitask Learners. Technical Report, OpenAI. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[44] Hinton, G., Vinyals, O. and Dean, J. (2015) Distilling the Knowledge in a Neural Network. arXiv:1503.02531.

[45] Sanh, V., Debut, L., Chaumond, J. and Wolf, T. (2019) DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. arXiv:1910.01108.

[46] Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., *et al.* (2024) Qwen2.5 Technical Report. arXiv:2412.15115.

[47] Schuster, M. and Nakajima, K. (2012) Japanese and Korean Voice Search. 2012 *IEEE International Conference on Acoustics*, *Speech and Signal Processing* (ICASSP), Kyoto, 25-30 March 2012, 5149-5152. https://doi.org/10.1109/icassp.2012.6289079

[48] Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., *et al.* (2016) Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. arXiv:1609.08144.

[49] Gage, P. (1994) A New Algorithm for Data Compression. *The C Users Journal*, **12**, 23-38.

[50] Sennrich, R., Haddow, B. and Birch, A. (2016) Neural Machine Translation of Rare Words with Subword Units. *Proceedings of the* 54*th Annual Meeting of the Association for Computational Linguistics* (Volume 1: Long Papers), Berlin, 7-12 August 2016, 1715-1725. https://doi.org/10.18653/v1/p16-1162

[51] Kudo, T. and Richardson, J. (2018) SentencePiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing. *Proceedings of the* 2018 *Conference on Empirical Methods in Natural Language Processing*: *System Demonstrations*, Brussels, 31 October-4 November 2018, 66-71. https://doi.org/10.18653/v1/d18-2012

[52] McCallum, A. and Nigam, K. (1998) A Comparison of Event Models for Naive Bayes Text Classification. Learning for Text Categorization: Papers from the 1998 AAAI Workshop, 41-48. https://aaai.org/papers/041-ws98-05-007/

[53] Genkin, A., Lewis, D.D. and Madigan, D. (2007) Large-Scale Bayesian Logistic Regression for Text Categorization. *Technometrics*, **49**, 291-304. https://doi.org/10.1198/004017007000000245

[54] Sebastiani, F. (2002) Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, **34**, 1-47. https://doi.org/10.1145/505282.505283

[55] Kim, Y. (2014) Convolutional Neural Networks for Sentence Classification. *Proceedings of the* 2014 *Conference on Empirical Methods in Natural Language Processing* (EMNLP), Doha, 25-29 October 2014, 1746-1751. https://doi.org/10.3115/v1/d14-1181

[56] Zhang, X., Zhao, J. and LeCun, Y. (2015) Character-Level Convolutional Networks for Text Classification. *Advances in Neural Information Processing Systems*, **28**, 649-657. https://proceedings.neurips.cc/paper/2015/hash/250cf8b51c773f3f8dc8b4be867a9a02-Abstract.html

[57] Hochreiter, S. and Schmidhuber, J. (1997) Long Short-Term Memory. *Neural Computation*, **9**, 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

[58] Liu, G. and Guo, J. (2019) Bidirectional LSTM with Attention Mechanism and Convolutional Layer for Text Classification. *Neurocomputing*, **337**, 325-338. https://doi.org/10.1016/j.neucom.2019.01.078

[59] Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N. and Huang, X. (2020) Pre-Trained Models for Natural Language Processing: A Survey. *Science China Technological Sciences*, **63**, 1872-1897. https://doi.org/10.1007/s11431-020-1647-3

[60] Trust, P. and Minghim, R. (2024) A Study on Text Classification in the Age of Large Language Models. *Machine Learning and Knowledge Extraction*, **6**, 2688-2721. https://doi.org/10.3390/make6040129

[61] Treviso, M., Lee, J., Ji, T., van Aken, B., Cao, Q., Ciosici, M.R., *et al.* (2023) Efficient Methods for Natural Language Processing: A Survey. *Transactions of the Association for Computational Linguistics*, **11**, 826-860. https://doi.org/10.1162/tacl_a_00577

[62] Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M. and Gelly, S. (2019) Parameter-Efficient Transfer Learning for NLP. *Proceedings of the* 36*th International Conference on Machine Learning*, **97**, 2790-2799. https://proceedings.mlr.press/v97/houlsby19a.html

[63] Li, X.L. and Liang, P. (2021) Prefix-Tuning: Optimizing Continuous Prompts for Generation. *Proceedings of the* 59*th Annual Meeting of the Association for Compu-*

tational Linguistics and the 11*th International Joint Conference on Natural Language Processing* (Volume 1: Long Papers), Online, 1-6 August 2021, 4582-4597. https://doi.org/10.18653/v1/2021.acl-long.353

[64] Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L. and Chen, W. (2022) LoRA: Low-Rank Adaptation of Large Language Models. *The Tenth International Conference on Learning Representations* (ICLR 2022), Virtual Event, 1-13. https://openreview.net/forum?id=nZeVKeeFYf9

[65] Dettmers, T., Pagnoni, A., Holtzman, A. and Zettlemoyer, L. (2023) QLoRA: Efficient Finetuning of Quantized LLMs. *Advances in Neural Information Processing Systems*, **36**, 10088-10115. https://doi.org/10.52202/075280-0441

[66] Zhang, Q., Chen, M., Bukharin, A., He, P., Cheng, Y., Chen, W. and Zhao, T. (2023) Adaptive Budg*et AL*location for Parameter-Efficient Fine-Tuning. *The Eleventh International Conference on Learning Representations* (ICLR 2023), Kigali, 1-17. https://openreview.net/forum?id=lq62uWRJjiY

[67] Ding, N., Qin, Y., Yang, G., Wei, F., Yang, Z., Su, Y., *et al.* (2023) Parameter-Efficient Fine-Tuning of Large-Scale Pre-Trained Language Models. *Nature Machine Intelligence*, **5**, 220-235. https://doi.org/10.1038/s42256-023-00626-4

[68] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., *et al.* (2011) Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, **12**, 2825-2830.

[69] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., *et al.* (2019) PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, **32**, 8024-8035.

[70] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., *et al.* (2020) Transformers: State-of-the-Art Natural Language Processing. *Proceedings of the* 2020 *Conference on Empirical Methods in Natural Language Processing*: *System Demonstrations*, Online, 16-20 November 2020, 38-45. https://doi.org/10.18653/v1/2020.emnlp-demos.6

[71] Greene, D. and Cunningham, P. (2006) Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering. *Proceedings of the* 23*rd International Conference on Machine Learning*, New York, 25-29 June 2006, 377-384. https://doi.org/10.1145/1143844.1143892

[72] Lang, K. (1995) NewsWeeder: Learning to Filter Netnews. In: Prieditis, A. and Russell, S., Eds., *Machine Learning Proceedings* 1995, Elsevier, 331-339. https://doi.org/10.1016/b978-1-55860-377-6.50048-7

[73] Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y. and Potts, C. (2011) Learning Word Vectors for Sentiment Analysis. *Proceedings of the* 49*th Annual Meeting of the Association for Computational Linguistics*: *Human Language Technologies*, Portland, 19-24 June 2011, 142-150. https://aclanthology.org/P11-1015/

[74] Farhangi, A., Sui, N., Hua, N., Bai, H., Huang, A. and Guo, Z. (2022) Protoformer: Embedding Prototypes for Transformers. In: Gama, J., Li, T., Yu, Y., Chen, E., Zheng, Y. and Teng, F., Eds., *Lecture Notes in Computer Science*, Springer International Publishing, 447-458. https://doi.org/10.1007/978-3-031-05933-9_35

[75] Lewis, D.D. (1987) Reuters-21578 Text Categorization Collection. UCI Machine Learning Repository [Dataset]. https://doi.org/10.24432/C52G6M

[76] Hersh, W., Buckley, C., Leone, T.J. and Hickam, D. (1994) OHSUMED: An Interactive Retrieval Evaluation and New Large Test Collection for Research. In: Croft, B.W. and van Rijsbergen, C.J., Eds., *Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*,

Springer, 192-201. https://doi.org/10.1007/978-1-4471-2099-5_20

[77] Clement, C.B., Bierbaum, M., O'Keeffe, K.P. and Alemi, A.A. (2019) On the Use of arXiv as a Dataset. arXiv:1905.00075.

[78] Lewis, D.D., Yang, Y., Rose, T.G. and Li, F. (2004) RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research*, **5**, 361-397. https://www.jmlr.org/papers/v5/lewis04a.html

[79] Meta (2024) Llama 3.2 Model Card. Covers Llama 3.2 1B and 3B. https://github.com/meta-llama/llama-models/blob/main/models/llama3_2/MODEL_CARD.md https://huggingface.co/meta-llama/Llama-3.2-1B

[80] Li, S., Zhao, Y., Varma, R., Salpekar, O., Noordhuis, P., Li, T., *et al.* (2020) PyTorch Distributed: Experiences on Accelerating Data Parallel Training. *Proceedings of the VLDB Endowment*, **13**, 3005-3018. https://doi.org/10.14778/3415478.3415530

[81] Wilcoxon, F. (1945) Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, **1**, 80-83. https://doi.org/10.2307/3001968

[82] Friedman, M. (1937) The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, **32**, 675-701. https://doi.org/10.1080/01621459.1937.10503522

[83] Dror, R., Baumer, G., Shlomov, S. and Reichart, R. (2018) The Hitchhiker's Guide to Testing Statistical Significance in Natural Language Processing. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics* (Volume 1: Long Papers), Melbourne, 15-20 July 2018, 1383-1392. https://doi.org/10.18653/v1/p18-1128

[84] Ethayarajh, K. (2019) How Contextual Are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings. *Proceedings of the* 2019 *Conference on Empirical Methods in Natural Language Processing and the* 9*th International Joint Conference on Natural Language Processing* (EMNLP-IJCNLP), Hong Kong SAR, 3-7 November 2019, 55-65. https://doi.org/10.18653/v1/d19-1006

[85] Reimers, N. and Gurevych, I. (2019) Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks. *Proceedings of the* 2019 *Conference on Empirical Methods in Natural Language Processing and the* 9*th International Joint Conference on Natural Language Processing* (EMNLP-IJCNLP), Hong Kong SAR, 3-7 November 2019, 3982-3992. https://doi.org/10.18653/v1/d19-1410

[86] Warner, B., Chaffin, A., Clavié, B., Weller, O., Hallström, O., Taghadouini, S., *et al.* (2025) Smarter, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference. *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics* (Volume 1: Long Papers), Vienna, 27 July-1 August 2025, 2526-2547. https://doi.org/10.18653/v1/2025.acl-long.127

[87] Tunstall, L., Reimers, N., Jo, U.E.S., Bates, L., Korat, D., Wasserblat, M. and Pereg, O. (2022) Efficient Few-Shot Learning without Prompts. arXiv:2209.11055.

A. Supplementary Results

A.1. Peak Micro-F1 by Family

Table A1. Peak Micro-F1 by family (as Table 5).

Corpus	Classical ML			Static-embedding NN			Transformer			Lead (pts)	
	F1	Clf	time	F1	Clf	time	F1	model/head	time	vs ML	vs NN
BBC News	0.989	SVM	0.85	0.989	CNN	7.46	0.992	RoBERTa/CNN	141	+0.3	+0.3
20 Newsgroups	0.715	LR	314	0.705	CNN	111	0.713	BERT/CNN	1876	−0.2	+0.8
IMDB	0.925	LR	198	0.891	ATTN	81.8	0.948	RoBERTa/SC	2366	+2.3	+5.7
ArXiv-10	0.857	LR	2569	0.830	ATTN	403	0.865	DeepSeek/SC	42,048	+0.8	+3.5
Reuters-21578	0.878	SVM	15.8	0.867	ATTN	241	0.899	XLNet/SC	6095	+2.1	+3.2
OHSUMED	0.706	SVM	13.2	0.714	CNN	194	0.770	DeepSeek/CNN	28,201	+6.4	+5.6
arXiv	0.785	LR	466	0.706	LSTM	89.2	0.826	DeepSeek/CNN	12,254	+4.1	+12.0
RCV1-v2	0.826	SVM	5519	0.803	ATTN	722	0.871	DeepSeek/CNN	96,232	+4.5	+6.8
Mean	0.835			0.813			0.861			+2.5	+4.7

A.2. Best Classical Configurations

Table A2. Feature set of the best classical run on each corpus (notation of Section 3.3; a model name denotes its mean-pooled frozen embeddings).

Corpus	Clf	Features	F1
*Macro-F*1			
BBC News	LR	TF-IDF ⊕ RoBERTa ⊕ Π (RoBERTa + WCE)	0.989
20 Newsgroups	SVM	Count ⊕ Llama	0.705
IMDB	LR	TF-IDF ⊕ RoBERTa	0.925
ArXiv-10	LR	TF-IDF ⊕ Llama	0.857
Reuters-21578	SVM	TF-IDF ⊕ fastText	0.606
OHSUMED	SVM	TF-IDF	0.691
arXiv	LR	Count ⊕ Llama	0.784
RCV1-v2	LR	Count ⊕ Llama	0.648
*Micro-F*1			
BBC News	SVM	TF-IDF	0.989
20 Newsgroups	LR	TF-IDF ⊕ Llama ⊕ Π (Llama + WCE)	0.715
IMDB	LR	TF-IDF ⊕ RoBERTa	0.925
ArXiv-10	LR	TF-IDF ⊕ Llama	0.857
Reuters-21578	SVM	TF-IDF ⊕ word2vec	0.878
OHSUMED	SVM	TF-IDF	0.706
arXiv	LR	Count ⊕ Llama	0.785
RCV1-v2	SVM	TF-IDF ⊕ RoBERTa	0.826

A.3. Validation Splits

Table A3. Training and validation sizes in the neural modules. Test sets are those of Table 1.

Corpus	Static-embedding NN		Transformer	
	Train	Validation	Train	Validation
BBC News	984	245	1014	215
20 Newsgroups	9052	2262	9335	1979
IMDB	20,000	5000	20,625	4375
ArXiv-10	66,000	16,500	68,063	14,437
Reuters-21578	7683	1920	7923	1680
OHSUMED	19,249	4812	19,851	4210
arXiv	9570	2392	9869	2093
RCV1-v2	18,520	4629	19,098	4051

A.4. Per-Classifier Results

Table A4 and Table A5 give, for each corpus and classifier, the maximum, median, mean and minimum score across runs, the number of runs and the median training time. Mean and run count match the corresponding statistics of [23].

Table A4. Macro-F1 by corpus and classifier.

Corpus	Clf	Max	Median	Mean	Min	Runs	Time (s)
BBC News	LR	0.989	0.980	0.969	0.692	186	3.00
	SVM	0.988	0.974	0.948	0.273	186	2.70
	NB	0.981	0.975	0.968	0.929	20	0.09
	CNN	0.988	0.982	0.979	0.959	12	7.71
	LSTM	0.955	0.927	0.930	0.909	12	5.70
	ATTN	0.967	0.944	0.947	0.928	12	5.33
	HF-SC	0.985	0.972	0.972	0.960	13	200
	HF-CNN	0.992	0.972	0.973	0.955	25	230
20 Newsgroups	LR	0.703	0.641	0.620	0.292	186	30.0
	SVM	0.705	0.625	0.590	0.058	186	27.3
	NB	0.641	0.595	0.586	0.433	20	10.0
	CNN	0.694	0.674	0.672	0.649	12	122
	LSTM	0.681	0.642	0.640	0.590	12	43.9
	ATTN	0.676	0.641	0.646	0.612	12	48.0
	HF-SC	0.706	0.684	0.683	0.663	13	2024
	HF-CNN	0.704	0.688	0.690	0.674	13	2257
IMDB	LR	0.925	0.857	0.855	0.686	190	45.2
	SVM	0.922	0.849	0.843	0.402	190	32.8
	NB	0.839	0.827	0.821	0.763	20	4.11
	CNN	0.891	0.869	0.870	0.853	12	167
	LSTM	0.887	0.870	0.867	0.832	12	121

Continued

	ATTN	0.891	0.884	0.883	0.864	12	102
	HF-SC	0.948	0.934	0.929	0.848	13	3258
	HF-CNN	0.948	0.935	0.889	0.333	13	2779
ArXiv-10	LR	0.857	0.821	0.810	0.601	244	140
	SVM	0.850	0.821	0.806	0.605	187	227
	NB	0.818	0.808	0.808	0.788	38	23.2
	CNN	0.828	0.806	0.804	0.764	12	293
	LSTM	0.828	0.824	0.822	0.804	11	328
	ATTN	0.832	0.823	0.823	0.807	12	373
	HF-SC	0.865	0.847	0.847	0.830	13	8633
	HF-CNN	0.864	0.847	0.848	0.839	13	9411
Reuters-21578	LR	0.600	0.526	0.510	0.203	186	30.0
	SVM	0.606	0.512	0.498	0.204	186	67.0
	NB	0.408	0.256	0.290	0.069	20	12.1
	CNN	0.576	0.532	0.536	0.486	12	126
	LSTM	0.535	0.484	0.481	0.411	12	261
	ATTN	0.592	0.553	0.552	0.489	12	192
	HF-SC	0.665	0.547	0.565	0.450	15	4260
	HF-CNN	0.659	0.606	0.596	0.492	21	4857
OHSUMED	LR	0.671	0.603	0.573	0.269	190	54.8
	SVM	0.691	0.585	0.567	0.270	190	174
	NB	0.510	0.287	0.309	0.123	20	12.8
	CNN	0.686	0.649	0.634	0.572	12	205
	LSTM	0.683	0.654	0.649	0.570	12	152
	ATTN	0.675	0.648	0.643	0.563	12	167
	HF-SC	0.745	0.712	0.715	0.676	13	5755
	HF-CNN	0.748	0.724	0.726	0.711	13	5001
arXiv	LR	0.784	0.652	0.624	0.208	186	31.8
	SVM	0.776	0.637	0.611	0.008	186	50.1
	NB	0.499	0.227	0.234	0.000	20	9.51
	CNN	0.691	0.595	0.595	0.459	24	132
	LSTM	0.696	0.568	0.499	0.000	23	130
	ATTN	0.694	0.581	0.579	0.483	24	126
	HF-SC	0.804	0.681	0.547	0.000	13	5174
	HF-CNN	0.821	0.726	0.731	0.708	13	2762
RCV1-v2	LR	0.648	0.492	0.495	0.218	171	496
	SVM	0.629	0.479	0.479	0.166	171	1592
	NB	0.381	0.164	0.221	0.113	18	7809
	CNN	0.498	0.421	0.435	0.398	4	544
	LSTM	0.535	0.487	0.478	0.403	5	761
	ATTN	0.550	0.504	0.508	0.468	8	717
	HF-SC	0.699	0.682	0.679	0.654	8	33,661
	HF-CNN	0.718	0.695	0.694	0.664	13	30,962

Table A5. Micro-F1 by corpus and classifier. One IMDB naive Bayes run has no logged Micro-F1 (Section 3.6), so that row has 19 runs and the table 3724.

Corpus	Clf	Max	Median	Mean	Min	Runs	Time (s)
BBC News	LR	0.989	0.981	0.970	0.697	186	3.00
	SVM	0.989	0.973	0.950	0.322	186	2.95
	NB	0.981	0.975	0.969	0.931	20	0.09
	CNN	0.989	0.983	0.981	0.962	12	7.71
	LSTM	0.958	0.929	0.934	0.916	12	5.70
	ATTN	0.969	0.946	0.950	0.935	12	5.33
	HF-SC	0.985	0.973	0.973	0.962	13	200
	HF-CNN	0.992	0.973	0.974	0.958	25	230
20 Newsgroups	LR	0.715	0.651	0.629	0.299	186	29.9
	SVM	0.711	0.631	0.596	0.078	186	27.3
	NB	0.671	0.624	0.610	0.426	20	10.0
	CNN	0.705	0.686	0.684	0.660	12	122
	LSTM	0.692	0.655	0.650	0.600	12	43.9
	ATTN	0.685	0.651	0.655	0.619	12	48.0
	HF-SC	0.708	0.695	0.693	0.677	13	2024
	HF-CNN	0.713	0.698	0.699	0.683	13	2257
IMDB	LR	0.925	0.857	0.855	0.686	190	45.2
	SVM	0.922	0.849	0.845	0.531	190	32.8
	NB	0.839	0.827	0.820	0.766	19	6.60
	CNN	0.891	0.869	0.870	0.853	12	167
	LSTM	0.887	0.870	0.867	0.832	12	121
	ATTN	0.891	0.884	0.883	0.864	12	102
	HF-SC	0.948	0.934	0.929	0.848	13	3258
	HF-CNN	0.948	0.935	0.902	0.500	13	2779
ArXiv-10	LR	0.857	0.821	0.810	0.607	244	140
	SVM	0.850	0.821	0.807	0.608	187	227
	NB	0.819	0.810	0.809	0.789	38	23.2
	CNN	0.826	0.806	0.803	0.764	12	293
	LSTM	0.827	0.824	0.822	0.803	11	328
	ATTN	0.830	0.823	0.822	0.805	12	373
	HF-SC	0.865	0.848	0.846	0.828	13	8633
	HF-CNN	0.865	0.848	0.849	0.837	13	9411
Reuters-21578	LR	0.860	0.819	0.799	0.406	186	30.0
	SVM	0.878	0.807	0.788	0.344	186	67.0
	NB	0.669	0.629	0.591	0.275	20	12.1

Continued

	CNN	0.856	0.837	0.840	0.823	12	125
	LSTM	0.850	0.816	0.813	0.763	12	261
	ATTN	0.867	0.837	0.842	0.822	12	192
	HF-SC	0.899	0.876	0.876	0.850	15	4260
	HF-CNN	0.897	0.887	0.886	0.868	21	4857
OHSUMED	LR	0.692	0.633	0.609	0.310	190	54.8
	SVM	0.706	0.616	0.602	0.310	190	174
	NB	0.562	0.391	0.422	0.298	20	12.8
	CNN	0.714	0.677	0.672	0.621	12	205
	LSTM	0.711	0.687	0.683	0.633	12	152
	ATTN	0.699	0.677	0.675	0.624	12	168
	HF-SC	0.764	0.736	0.739	0.720	13	5755
	HF-CNN	0.770	0.748	0.746	0.734	13	5001
arXiv	LR	0.785	0.655	0.620	0.195	186	31.8
	SVM	0.783	0.643	0.611	0.020	186	50.1
	NB	0.467	0.213	0.221	0.000	20	9.51
	CNN	0.706	0.617	0.615	0.492	24	132
	LSTM	0.706	0.585	0.510	0.000	23	130
	ATTN	0.700	0.596	0.591	0.493	24	126
	HF-SC	0.812	0.685	0.553	0.000	13	5174
	HF-CNN	0.826	0.733	0.739	0.709	13	2762
RCV1-v2	LR	0.819	0.750	0.727	0.362	171	496
	SVM	0.826	0.742	0.723	0.435	171	1592
	NB	0.618	0.575	0.558	0.358	18	7809
	CNN	0.800	0.714	0.730	0.692	4	544
	LSTM	0.802	0.782	0.773	0.733	5	761
	ATTN	0.803	0.770	0.769	0.728	8	717
	HF-SC	0.866	0.858	0.857	0.846	8	33,661
	HF-CNN	0.871	0.863	0.863	0.854	13	30,962