Task-Agnostic Web Document Annotation with LLM-Generated Descriptors

Anonymous ACL submission

Abstract

Web-scale text corpora provide NLP researchers and corpus linguists with access to vast quantities of language data, but efficiently identifying the subsets of a large corpus that are relevant to a particular research aim remains a significant challenge. Existing approaches, including keyword search and task-specific classifiers, are often either too restrictive or depend on costly retraining for each new research question. We explore and evaluate the feasibility of a task-agnostic, LLM-driven descriptor schema for annotating web documents, using LLMs to generate natural-language labels that capture stylistic, tonal, topical, and structural properties of texts. Unlike prior work that emphasizes broad categorization or quality filtering, our approach yields fine-grained and interpretable descriptors designed to support flexible downstream applications. We demonstrate the approach through two use cases, showing that descriptor-based annotation can enable flexible document retrieval while also exposing challenges related to schema harmonization and descriptor specificity.

1 Introduction

Natural language corpora have grown in tandem with computational resources. The 1-million-word Brown corpus built in the 1960s is tiny compared to the corpora used today, such as the 15-trillion-token FineWeb (Penedo et al., 2024a). Large web corpora provide unprecedented insight into language use on the internet and are used to train state-of-the-art Large Language Models (LLMs). To efficiently use these resources, researchers have to characterize, categorize, and filter them for various downstream applications. As these datasets continue to grow in size, this is becoming increasingly challenging.

Large corpora are often only the starting point for a research project and are rarely used in their entirety. Corpus linguists typically only focus on a small subset of documents that are relevant to their research questions. Even for LLM training, requiring trillions of tokens, identifying the highest quality documents can be more important than focusing on quantity (Su et al., 2025; Lozhkov et al., 2024; Li et al., 2024; Diao et al., 2025).

In order to facilitate broader adoption of these vast collections in research, the data need rich annotations and metadata. In this paper, we build and evaluate a data-driven task-agnostic approach for generating open-ended document annotations. We leverage LLMs to generate a comprehensive document descriptor schema with more than 800,000 unique labels that can be used to create subsets of data for many purposes with relative ease. We test the applicability of the descriptors in different research use cases to evaluate their reusability for different tasks and settings. We present a proof-of-concept implementation of the approach and the steps taken in developing it. We release our code and the descriptor schema, which can be applied to any English corpus of web documents. Additionally, we release a ready-to-use 10-billion-token descriptor-annotated corpus.

2 Background and related work

The idea of using internet as a source of large-scale language data begun at the start of the millennium with the Web-as-Corpus approach (Kilgarriff and Grefenstette, 2003). The value of metadata in large web corpora has also long been recognized (e.g., Barbaresi 2016) and, in recent years, LLMs have emerged as tools for annotating and analyzing text data at scale (Tan et al., 2024; Yadav et al., 2024; Chen et al., 2024a,b). ChatGPT and other state-of-the-art LLMs already outperform human coders in many annotation tasks (e.g., Bermejo et al. (2024); Gilardi et al. (2023); Törnberg (2024)). For example, Maragheh et al. (2023) use an LLM to generate thematic keywords for English news and e-commerce documents while Zhang and Takada (2025) use GPT-4o in an active learning setting,

where documents are labeled with the LLM for classification tasks.

Identifying valuable documents for LLM training from large web corpora has become increasingly important (Penedo et al., 2024a; Chen et al., 2024a; Su et al., 2025; Li et al., 2024). To this end, Wettig et al. (2025) introduce WebOrganizer, two parallel taxonomies that characterize the topic and format of web documents. They use Llama models to annotate documents and train smaller classifiers for subsequent large-scale annotation. Penedo et al. (2024b) use an LLM to annotate web documents for their educational value and train a classifier to produce FineWeb-Edu. Idahl et al. (2026) use an LLM to annotate web documents along 18 quality axes such as completeness, structure, and information density.

All of the approaches above follow the same basic steps: (1) Annotate data manually or with an LLM. (2) Train a classifier on the annotated data. (3) Run inference on the full dataset. Each of these steps requires considerable compute effort, since the corpus needs to be labeled, as well as extensive human labor if the training data is created manually. These approaches rely on manually crafted, task-specific annotation schemas and categorize documents into a relatively small number of broadly defined categories with coarse labels. This loses detail, limiting the usability of the annotations to applications directly supported by the annotation schema. For instance, if one were to seek to retrieve "sarcastic" documents in the presently available large web corpora, no metadata in these corpora allows that query to be made. Consequently, the cycle of annotation, training, and inference on a large corpus must be repeated for this, and every other new task not directly supported by the existing annotation schemas.

Rather than create annotations separately for each use case, the aim of this paper is to explore an approach whereby each document is given a task-agnostic annotation that is detailed enough to support a wide variety of tasks. In this approach, an LLM is used to annotate the data only once and then relies on the same annotations for all tasks. This is achieved through a bottom-up approach in which the LLM generates a labeling schema spanning hundreds of thousands of unique labels instead of relying on a handful of predetermined categories.

descriptor	
tag	**definition**
formal tone and language	The formal tone and language used convey credibility, authority, and expertise, making it suitable for professional or official contexts.

Figure 1: Example of a descriptor, consisting of a tag and a definition

3 Method

Our objective is to assign each document a number of *descriptors*, where each descriptor captures some aspect of the document. Every descriptor consists of two parts: a unique *tag* and a *definition* (see Figure 1). The tag is a short phrase (usually 1–3 words) that holds the main meaning, while the definition expands on it by providing an interpretable explanation with additional context and specificity.

3.1 The descriptor schema

As the basis of our descriptor schema, we use a sample of 500,000 documents from FineWeb, a collection of relatively high-quality English documents, which we label using Llama-3.3-70B-Instruct (Grattafiori et al., 2024). In initial experiments, we found that this model generates high-quality descriptors while being computationally manageable at our data sizes.[1]

The annotation is carried out in an open-ended fashion, with descriptors created dynamically during the annotation process. Since each document is labeled independently, a major methodological challenge is ensuring that the descriptors are detailed enough to cover all aspects of the document while avoiding semantic duplication, where the same phenomena are annotated with differently worded descriptors. The most straightforward approach would be iterative, whereby the model is informed about descriptors assigned in earlier data batches in the prompt and instructed to introduce new descriptors only when necessary. To examine this strategy, we conducted a preliminary experiment using tags alone, on the assumption that the model would find it easier to track existing tags than full descriptors. Figure 2 shows the growth of the tag vocabulary under three prompting conditions: providing the full, growing tag vocabulary, providing no vocabulary, or providing only the top N most common prior tags.

[1]While newer models have become available since our initial runs, re-running all experiments is unlikely to produce radically different results to justify the resource cost.

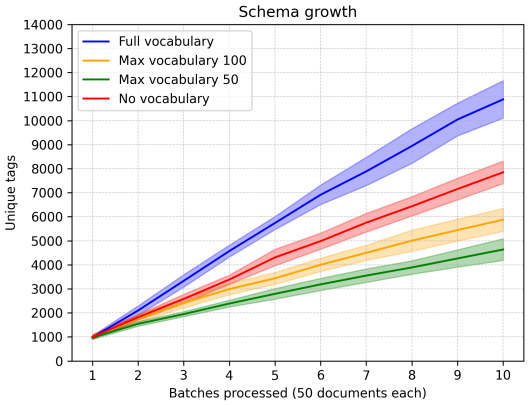

Figure 2: Rate of tag schema growth with different maximum vocabularies (average of 5 runs). Vocabulary size refers to the number of existing tags given in the prompt.

Our first finding is that the model struggles to adhere to the given schema. We observed substantial levels of synonymous tags in the model output even when the model was informed of suitable existing tags, indicating that it often introduced new tags unnecessarily for concepts already covered by the schema. Moreover, contrary to expectations, providing a larger vocabulary accelerated the rate at which new tags were generated. This effect is shown in Figure 2, where the full vocabulary yielded the fastest growth rate.

Since generating the schema and labeling the documents in a single pass was found to be unfeasible, an alternative approach is to first generate descriptors freely and subsequently refine the schema by merging duplicates and synonyms. This results in a three-step pipeline: (Step 1) Generation: the initial descriptors are generated for each document independently; (Step 2) Disambiguation: descriptors with duplicate tags are either merged, with their definitions revised accordingly, or renamed so that each unique descriptor has a unique tag; (Step 3) Synonym merging: synonymous or near-synonymous descriptors are merged. The aim of the disambiguation and synonym merging steps is to produce a set of descriptors where each unique tag has exactly one definition and the prevalence of synonymous or near-synonymous descriptors is minimal.

To preserve space, the descriptions of the steps in the following sections are abbreviated. For a more comprehensive description and the full prompts used, refer to Appendices A and B. We also discuss the steps taken in developing the pipeline in more detail in Section 5.

3.1.1 Generation step

The generation step consists of three stages: generation, rewriting, and revision. First, the model is given one document and is prompted to "[d]escribe the document in aspects including, but not limited to, style, tone, genre, topic, domain, language, quality, sentiment, etc." For each tag, the model is also prompted to generate a definition, which should "explain its meaning and relevance to the document in 2 to 3 sentences". The generated descriptors should be detailed enough to support approximate reconstruction of the original content from the descriptors alone. However, to avoid mixing highly specific details with more general features, the model is prompted to place minute details under *specifics*. These are used to evaluate the model's ability to reconstruct the document, but are discarded from the remaining analyses.

To evaluate how well the meaning of the document is captured in the descriptors, we give the descriptors generated in the previous stage back to the model and ask it to rewrite the corresponding document. If the generated text closely resembles the original, this suggests that the descriptors have preserved its meaning.

Since the LLM might not generate high-quality descriptors in its first attempt, the model is asked to evaluate the quality of the rewrite and revise the descriptors accordingly. The model compares the original document to its rewrite and generates a revised set of descriptors that better capture the meaning of the document. The revision stage is followed by another rewrite, now using the revised descriptors, and then another revision. Although this process could be repeated arbitrarily many times, we stop after three sets of descriptors and rewrites to preserve computational resources.

To identify the set of descriptors that best captured the meaning of the original document, we embed the original document and each of its rewrites using the Stella-en-400M-v5 model (StellaEncoder, 2024) and calculate their cosine similarity. The closer a rewrite is to the original document, the better the descriptors have summarized the document and thus, the better their quality. Our experiments confirm that rewrite similarity increases with more descriptor revisions (Figure 3).

After these steps, only the best descriptors for each document are kept. Labeling our sample results in nearly 7.5 million unique descriptors. Each

document gets an average of 15 descriptors and some of the most common tags are “Formal tone”, “Conversational tone” and “Technical”.

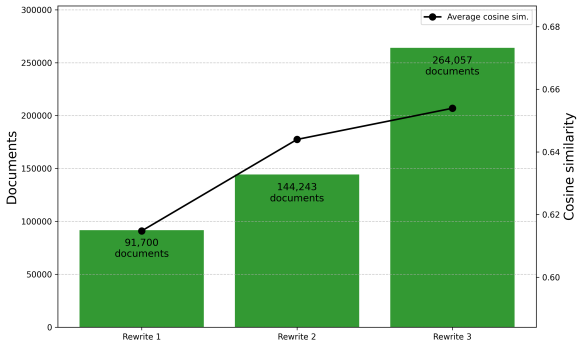

Figure 3: Number of documents whose best rewrite was generated on the first, second and third attempt. While the third round most often generated the best rewrite, increase in cosine similarity between attempts was modest on average.

3.1.2 Disambiguation step

In the disambiguation step, we normalize all tags to lowercase and group together all definitions for a given tag. The vast majority of tags have only one definition, but some have more, with the most common ones having thousands of different definitions. The groups with more than one definition are given to the LLM, which is prompted to either merge together (near-)synonymous entries and provide one comprehensive descriptor, or split the group into several descriptors, each consisting of a locally unique tag and definition.

3.1.3 Synonym merging step

After disambiguation, the schema still contains many (near-)synonymous descriptors because only exact tag duplicates were grouped together in the previous step. For instance, tags such as "formal tone" and "formal language" are still separate at this point. In the synonym merging step, each descriptor is first embedded using the Stella model, and mutually nearest neighbors are provided to the LLM. The model either accepts them as synonyms and chooses one to represent both or rejects their synonymity and keeps them as separate entries. After each iteration, new mutually nearest neighbors are identified and provided to the model until none are left. After synonym merging, the descriptor schema has condensed from nearly 7.5 million to around 0.8 million unique descriptors, where each tag is unique and has only one definition (Table 1).

Step	Descriptors
Generation	7,433,036
Disambiguation	1,001,994
Synonym merging	800,488

Table 1: Number of unique descriptors after each step in the pipeline.

3.2 Harmonization

The previous three steps, when carried out on a sufficiently large number of documents, will result in a schema of descriptors capable of describing a broad variety of documents. Any new documents can thereafter be annotated in this schema, without further updates. For new documents, we first apply the descriptor generation step (Section 3.1.1). This produces document-specific descriptors that do not necessarily correspond to the schema. We refer to these independently generated descriptors as *raw* descriptors throughout the remainder of the paper.

Next, these raw descriptors are mapped onto the existing schema. We refer to the resulting schema-mapped descriptors as *harmonized* descriptors. For each raw descriptor, we retrieve the 10 most similar descriptors from the schema[2] and use the Llama model to select the best match among them. If the model rejects all 10 candidates, the descriptor is dropped. In the end, each document has a set of raw descriptors and a similar but schema-adhering set of harmonized descriptors.

4 Evaluation and use cases

In this section, we evaluate the use of descriptors for sub-corpus selection. First, we evaluate how accurately the descriptors describe their respective documents by using a judge LLM. Then, we present and evaluate two use cases for document selection using the descriptors. We do all evaluation on both raw and harmonized descriptors to estimate the effect of the harmonization process.

Some of the evaluations require a reference classification of the documents to which the descriptors can be compared. One such is the FineWeb-Edu subset, which assigns an *educational* label to 8% of FineWeb documents (Lozhkov et al., 2024). We preserve this proportion when sampling our test dataset. As the second source of reference classification, we apply the WebOrganizer format and

[2] Based on cosine similarity of their embeddings calculated using the Stella model.

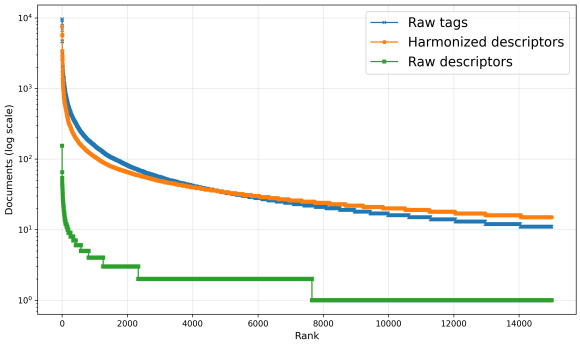

Figure 4: The number of documents in which the 15,000 most common tags and descriptors occur in the harmonized and raw setting. Harmonized tags not shown since their counts equal harmonized descriptors.

topic classifiers, which assign each document one format and one topic from the WebOrganizer taxonomy (Wettig et al., 2025).

As test data, we use a sample of 100,000 FineWeb documents that were not used to derive the descriptor schema. On average, each document in the test set received 15 descriptors. Nearly all raw descriptors appear in only one document, while the average harmonized descriptor appears in nearly 5 documents. The distribution is shown in Figure 4.

4.1 Evaluating descriptor accuracy

We evaluate how well the given descriptors describe their corresponding documents. This question is especially meaningful for harmonized descriptors, as some loss in accuracy is almost inevitable when a raw descriptor is replaced with its (near-)synonym from the schema. We evaluate the accuracy of the descriptors on a sample of 5,000 documents from the test data. We use Qwen3-Next-80B-A3B-Instruct (Qwen Team, 2025) as a judge model. The judge model is given a document and its descriptors and is prompted to evaluate how well the descriptors describe the document on a five-point scale. The results are summarized in Table 2. Almost 90% of the raw descriptors are judged to be accurate, while only approximately 5% are inaccurate or mostly inaccurate. As expected, the harmonization process costs accuracy. Still, nearly 80% of harmonized descriptors are judged to be at least partially accurate.

4.2 Use case 1: Finding documents by query

As the first use-case test, we evaluate descriptor-based document retrieval for a given query, simulating a corpus user looking for a subset of documents defined by a property described in a short query. We evaluate this approach on three document types: "FAQ", "legal notice", and "sarcastic". The first two categories are format labels derived from the WebOrganizer format classifier. The third category represents a text type that is difficult to identify using traditional query methods, as it cannot be identified by any particular keywords.

Model judgment	Raw	Harmonized
Accurate	88.67%	61.81%
Mostly accurate	0.83%	2.90%
Partially accurate	5.58%	14.61%
Mostly inaccurate	0.55%	1.51%
Inaccurate	4.36%	19.16%

Table 2: Descriptor accuracy as determined by a judge model.

To retrieve documents corresponding to the queries, we apply the following procedure: (1) Format each query as a descriptor consisting of a tag and a definition. (2) Retrieve the top K matching descriptors for the query from among the descriptors present in the test data using embedding similarity with the Stella model and the Faiss index (Douze et al., 2026). K is intentionally selected for high recall, i.e., many of the retrieved descriptors will be irrelevant to the query. (3) Use the judge LLM to retain only those retrieved descriptors that are relevant to the given query. These then form a set of query descriptors used to retrieve the matching documents. The numbers of descriptors returned by the initial similarity search and after LLM pruning are summarized in Table 3.

	Raw		Harmonized	
Target	Faiss	LLM	Faiss	LLM
FAQ	644	380	76	22
Legal Notices	7019	1579	1282	242
Sarcasm	18045	5484	1032	277

Table 3: Numbers of descriptors returned with initial similarity search and the number of descriptors accepted by the judge model.

Finally, the LLM judge evaluates whether the documents associated with the approved descriptors correspond to the original query. Because many documents may correspond only partially to the query, for instance by containing a legal notice or FAQ section at the end of the document, the judge is given three response options: "Yes", "No", and "Partial", indicating full, no, and partial

correspondence, respectively.

4.2.1 Results for use case 1

The FAQ query retrieved 255 documents with 22 harmonized descriptors, while 380 raw descriptors matched 380 documents. For legal notices, 242 harmonized descriptors returned 994 documents, and 1,579 raw descriptors returned 907 documents.

The WebOrganizer format classifier assigns only one label per document, whereas there are on average 15 descriptors per document. For instance, documents that contain an FAQ or legal notice as a subsection will not receive the corresponding WebOrganizer format tag, but will receive the corresponding descriptor. This makes direct comparison between the two annotations difficult. When using labels given by the WebOrganizer format classifier as the ground truth, FAQ retrieval obtains an F-score of 0.28–0.30 and legal notice retrieval an F-score of 0.36–0.37 (Table 4). FAQ retrieval has higher precision, whereas with legal notices we achieve higher recall.

Overall, the harmonized descriptors provide broader coverage with a smaller descriptor set, but at the cost of lower accuracy. By contrast, the raw descriptors produce fewer results but with higher accuracy. However, these scores should be interpreted with caution due to the differences in labeling approaches. We return to this issue in the error analysis (Section 4.4).

Label	Type	Prec.	Rec.	F1
Legal Notices	Harmonized	0.30	0.48	0.37
Legal Notices	Raw	0.31	0.45	0.36
FAQ	Harmonized	0.38	0.22	0.28
FAQ	Raw	0.32	0.28	0.30

Table 4: Precision, recall, and F1 scores for legal notices and FAQs under harmonized and raw settings.

For the sarcasm query, the harmonized and raw descriptors retrieved 5,459 and 3,821 documents, respectively. Among the documents retrieved with harmonized descriptors, the judge model classified 67.9% as full matches to the query and 6.2% as partial matches. For documents retrieved with raw descriptors, the rates were 79.7% and 5.4%, respectively.

As Figure 5 shows, descriptor acceptance generally declines with decreasing descriptor–query similarity. However, documents retrieved through LLM-approved descriptors are accepted at high rates, indicating that the approved descriptors are reliable indicators of sarcasm. The lower and more variable document-level acceptance rates for harmonized descriptors reflect their broader coverage: a single harmonized descriptor often retrieves many documents, including some irrelevant ones. Overall, the results suggest that LLM-approved descriptors are effective for identifying sarcastic documents, especially at higher similarity thresholds.

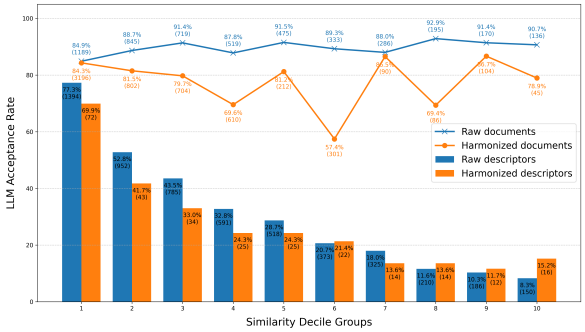

Figure 5: LLM acceptance rates for descriptors and documents retrieved by the sarcasm query, by descriptor–query similarity decile. Bars show the share of retrieved descriptors accepted by the LLM judge; lines show the share of associated documents accepted by the judge. Counts of accepted descriptors/documents are in parentheses. Document counts may exceed the number of unique retrieved documents because a document can be retrieved multiple times by different descriptors.

4.3 Use case 2: Descriptors as features for logistic regression

In use case 2, we treat the descriptors as features for training logistic regression classifiers. The aim is to test whether the descriptors can serve as a cheap filter for a corpus, ranking documents to measure how much of the corpus must be scanned to recover a target fraction of each class.

We split the data into 80/20 training and test sets and target three sets of predicted target labels: the binary FineWeb-Edu label and the WebOrganizer topic and format taxonomies, each with 24 classes. For each task, we train a one-vs-rest logistic regression model. The computational cost is minimal: using the full harmonized descriptor vocabulary (278,093 features) over the 80,000-document training set, training all three models took 70 seconds on a single CPU core, and inference over the 20,000-document test set took 27 ms in total. We use the *tags* (not full descriptors) as features, encoded as a binary presence vector; in the harmonized setting each tag has a unique definition, so harmonized tags and descriptors are equivalent. For reference,

we also train TF-IDF models (1–2-grams) from the full texts. We rank the test documents by the model's score for each class and report the fraction of the test corpus that must be scanned to reach 80% and 20% recall. The full per-class retrieval results appear in Appendix E (Tables 6–11).

4.3.1 Results for use case 2

Comparing against TF-IDF capped at 50,000 features (Table 5), the raw descriptors are competitive on macro-F1 on the educational label (0.76 vs. 0.75) and on format (0.48 vs. 0.49), but behind on topic (0.65 vs. 0.73). Raw descriptors outperform harmonized ones on all three tasks, reflecting the accuracy loss from harmonization (Section 4.1). The descriptors are also more interpretable as features: most highest-weighted features are interpretable phrases like "scientific-research" and "educational content", whereas TF-IDF's are bare tokens like "or" and "of the" (Table 12). Since only the tag is used, some are vaguer in isolation (e.g. "general context"). The descriptors also behave differently from TF-IDF as the vocabulary grows (Figure 7). On the educational and format tasks, TF-IDF peaks early and declines, while the descriptors improve and level off; on topic both improve and flatten, and TF-IDF does not decline.

As a ranking-based pre-filter, the descriptors are weaker than TF-IDF but still effective. To reach 80% recall on topic, the descriptors require scanning 11.6–11.8% of the corpus against 7.2% for TF-IDF, and on format 22–24% against 16%. The gap narrows considerably at the top of the ranking: at 20% recall the descriptors scan nearly the same fraction as TF-IDF (1.3% vs. 1.1% on topic, 2.6–2.7% vs. 2.2% on format). Performance also varies by class: the descriptors are strongest on semantically coherent categories such as Adult, Religion, and Legal Notices, where they are competitive with TF-IDF, and weakest on classes defined by surface form or boilerplate, such as FAQ and Content Listing. Overall, for most classes the descriptors recover the majority of positive documents while scanning only a small fraction of the corpus.

4.4 Error analysis

Some errors in use case 1 seem to result from the classifier assigning only one label per document and, therefore, missing documents that contain an FAQ or legal notice only at the end. By contrast, the descriptors can describe specific parts of a document and may therefore be more accurate at identifying these kinds of mixed-format documents. This interpretation is supported by the judge model, which shows that many mixed-format document were not found with WebOrganizer. We also notice that the WebOrganizer classifier often classifies documents with any legal jargon as legal notices, even when they are not legal notices as defined in the WebOrganizer paper (Wettig et al., 2025, p. 18).

Beyond mixed-format documents, some false positives stem from tenuous query–descriptor matches, or from descriptors covering only a small part of the document. For example, a long document with a one-line copyright statement should arguably not count as a legal notice, yet it might receive a corresponding descriptor. False negatives often occur when the descriptors use vague terms or terms that differ from the query. For instance, FAQ documents may be tagged simply as "Informative" or "Informational". In some cases, the meaning of the query term is distributed across multiple descriptors, none of which alone is sufficient to be a good match. Adjusting how descriptors are queried could help alleviate these issues.

5 Developing the descriptor pipeline

In this section, we discuss how the descriptor generation pipeline was developed, providing context and explanations for the decisions we made. As mentioned in Section 3.1, we initially tried to include the full schema in the model prompt. Ideally, the annotation schema would grow quickly at first and then plateau, as most of the required labels would already exist. This approach would generate a comprehensive labeling schema in a single pass through the data. However, we found that the model did not adhere to the given schema. Additionally, the longer model input had a major impact on processing speed.

Definitions were added to descriptors for two reasons. First, without definitions, the meaning of the descriptors would be too vague. For example, a document with dark humor and one with light jokes could both be labeled "Humorous". With a definition, the type of humor can be specified. Second, adding detail and length leads to better embeddings, which is useful for downstream tasks.

Another issue we faced was that the model tended to quote the document directly or generate overly specific descriptors. Therefore, we instruct the model to place these *specifics* in a separate list from the descriptors. The specifics are used only

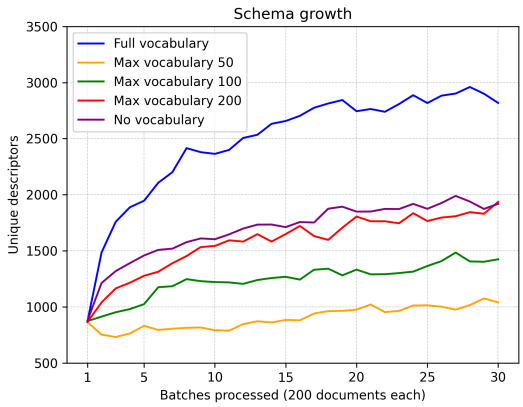

Figure 6: Rate of schema growth with different maximum vocabularies with synonym merging after each batch. Vocabulary size refers to the number of labels given to the model in the prompt.

in the initial descriptor generation step, after which they are discarded.

The final stage of development was merging. In our first merging implementation, we clustered and merged new descriptors with the existing schema after each new batch of data. This slowed the growth of the descriptor schema (see Figure 6). However, descriptors that were lexically similar but semantically opposed, such as "Formal" and "Informal", tended to get merged. To address this issue, the LLM was tasked with deciding whether and how to merge a cluster. While this approach worked better, it slowed descriptor generation drastically, especially as the number of descriptors grew. Hence, merging was ultimately implemented in two separate post-processing steps, as described in Sections 3.1.2 and 3.1.3.

6 Discussion and conclusions

In this paper, we have explored a new approach to enriching web corpora with metadata that is, on the one hand, human-readable and lends itself to straightforward querying and, on the other hand, sufficiently detailed to support a broad spectrum of tasks without the need to specify them *a priori*. The main motivation of this study is to enable the categorization and retrieval of task-specific subsets of large web corpora without having to apply the resource-intensive annotate–train–classify approach separately to each new task. Crucially, in our approach, the effort of assigning the descriptors needs to be spent only once.

Our first finding is that LLMs can assign detailed, human-readable, and useful descriptors to documents, even without an a priori schema. We demonstrate this in three ways. First, an LLM-as-judge evaluation shows that individual descriptors accurately describe the documents (Section 4.1). Second, we show that a simple and fast query procedure can retrieve a non-trivial number of documents matching FAQs, legal notices, and sarcasm; phenomena that would be difficult to identify in large corpora without a fine-tuned classifier (Section 4.2). This simulates a researcher identifying a task-specific subset from a larger corpus using a query not known when the descriptor schema was created. Third, we show that descriptors can serve as features for a simple logistic regression model that filters out the vast majority of irrelevant documents, reducing the cost of applying a more powerful classifier to the remainder (Section 4.3). This experiment simulates classifier-based pre-filtering of large web corpora. Compared with a TF-IDF classifier trained on full document texts, the descriptors faithfully capture document contents for several, though not all, tasks.

Our second finding is that LLMs are currently unable to be sufficiently systematic across documents. Our attempts to devise the descriptor schema and annotate the documents in a single pass were unsuccessful. We were able to alleviate, although not fully solve (Table 2), the problem by disambiguating and merging the descriptors post hoc using a combination of embedding similarity and LLM judgments. Further research is needed to develop a method that would allow an LLM to devise and follow a large descriptor schema.

To conclude, we strive to create rich, human-readable metadata for large corpora, produced only once and applicable to any number of tasks without the costly training and application of task-specific document classifiers. Such metadata could improve the accessibility of large corpora to new groups of users seeking to extract very specific subsets of documents.

The pipeline and evaluation code are available at `https://anonymous.4open.science/r/web-document-descriptors-2BBB`. We will release a 10T token sample of the FineWeb dataset annotated with the descriptors. An interactive demo allowing descriptor querying on a small sample of documents is available at `https://anonymous.4open.science/r/descriptor_demo_fork-F51D`.

Limitations

This research is limited to monolingual English documents from the FineWeb dataset, which is purposefully built to exclude low-quality documents. As a result, the descriptor schema derived from it may not generalize well to noisier datasets. Additionally, the findings do not yet show whether the approach generalizes to other languages or multilingual settings. Future work should therefore evaluate the method on multilingual corpora.

A second limitation concerns the language model used for descriptor generation and harmonization. Since LLM capabilities continue to improve rapidly, newer models may produce more accurate, consistent, and multilingual descriptors. Consequently, the results reported here should be understood as reflecting the performance of one specific model rather than the upper bound of the proposed method.

Finally, although applying existing descriptors to downstream tasks is relatively fast and inexpensive, generating descriptors for large datasets remains computationally costly. To address this limitation, future work could explore smaller fine-tuned models that generate descriptors more efficiently while preserving descriptor quality. We plan to train and release such a model in future work.

Ethical considerations

This research relies on generative LLMs which are known to produce errors in their output. Researchers should take care to evaluate the descriptor schema themselves before applying it to their own research. While we use a high-quality dataset, FineWeb, as the basis of the descriptor schema, the documents might still contain biased, racist, sexist or otherwise unreasonable or harmful content, which may be reproduced in the document descriptors. Using LLMs consumes electricity, which, depending on the production method, can lead to greenhouse gas emissions. For this paper, processing 500,000 documents through the pipeline required an estimated 1,000 GPU hours on four AMD MI250X GPUs and all electricity came from renewable resources.

AI assistants were used to proofread sections of the paper. Some of the code used in the experiments was written with AI assistance. All AI generated text and code was reviewed and verified by the authors.

References

Adrien Barbaresi. 2016. Efficient construction of metadata-enhanced web corpora. In *Proceedings of the 10th Web as Corpus Workshop*, pages 7–16, Berlin. Association for Computational Linguistics.

Vicente J. Bermejo, Andres Gago, Ramiro H. Gálvez, and Nicolás Harari. 2024. Llms outperform outsourced human coders on complex textual analysis. SSRN Working Paper.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and Hongxia Jin. 2024a. Alpagasus: Training a better alpaca with fewer data.

Ruirui Chen, Chengwei Qin, Weifeng Jiang, and Dongkyu Choi. 2024b. Is a large language model a good annotator for event extraction? *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17772–17780.

Shizhe Diao, Yu Yang, Yonggan Fu, Xin Dong, Dan Su, Markus Kliegl, Zijia Chen, Peter Belcak, Yoshi Suhara, Hongxu Yin, Mostofa Patwary, Yingyan, Lin, Jan Kautz, and Pavlo Molchanov. 2025. Nemotron-climb: Clustering-based iterative data mixture bootstrapping for language model pre-training.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2026. The faiss library. *IEEE Transactions on Big Data*, 12(2):346–361.

Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. 2023. ChatGPT outperforms crowd workers for text-annotation tasks. *Proceedings of the National Academy of Sciences*, 120(30).

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan

Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma,

Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models.

Maximilian Idahl, Benedikt Droste, Björn Plüster, and Jan Philipp Harries. 2026. propella-1: Multi-property document annotation for llm data curation at scale.

Adam Kilgarriff and Gregory Grefenstette. 2003. Introduction to the special issue on the web as corpus. *Computational Linguistics*, 29:333–347.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Abbas, Cheng-Yu Hsieh, Dhruba Ghosh, Josh Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Chandu, Thao Nguyen, Igor Vasiljevic, Sham Kakade, Shuran Song, Sujay Sanghavi, Fartash Faghri, Sewoong Oh, Luke Zettlemoyer, Kyle Lo, Alaaeldin El-Nouby, Hadi Pouransari, Alexander Toshev, Stephanie Wang, Dirk Groeneveld, Luca Soldaini, Pang Wei Koh, Jenia Jitsev, Thomas Kollar, Alexandros G. Dimakis, Yair Carmon, Achal Dave, Ludwig Schmidt, and Vaishaal Shankar. 2024. Datacomp-lm: In search of the next generation of training sets for language models.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. 2024. Fineweb-edu: the finest collection of educational content.

Reza Yousefi Maragheh, Chenhao Fang, Charan Chand Irugu, Parth Parikh, Jason Cho, Jianpeng Xu, Saranyan Sukumar, Malay Patel, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2023. Llm-take: Theme-aware keyword extraction using large language models. In *2023 IEEE International Conference on Big Data (BigData)*, pages 4318–4324.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2023. Mteb: Massive text embedding benchmark.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, et al. 2024a. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024b. The fineweb datasets: Decanting the web for the finest text data at scale.

Qwen Team. 2025. Qwen3 technical report.

StellaEncoder. 2024. stella_en_400m_v5.

Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. 2025. Nemotron-cc: Transforming common crawl into a refined long-horizon pretraining dataset.

Zhen Tan, Dawei Li, Song Wang, Alimohammad Beigi, Bohan Jiang, Amrita Bhattacharjee, Mansooreh Karami, Jundong Li, Lu Cheng, and Huan Liu. 2024. Large language models for data annotation and synthesis: A survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 930–957, Miami, Florida, USA. Association for Computational Linguistics.

Petter Törnberg. 2024. Large language models outperform expert coders and supervised classifiers at annotating political social media messages. *Social Science Computer Review*.

Alexander Wettig, Kyle Lo, Sewon Min, Hannaneh Hajishirzi, Danqi Chen, and Luca Soldaini. 2025. Organize the web: Constructing domains enhances pre-training data curation.

Sachin Yadav, Tejaswi Choppa, and Dominik Schlechtweg. 2024. Towards automating text annotation: A case study on semantic proximity annotation using gpt-4.

Yejian Zhang and Shingo Takada. 2025. Applying llms to active learning: Toward cost-efficient cross-task text classification without manually labeled data. *International Journal of Intelligent Systems*, 2025(1):6472544.

A Comprehensive description of the descriptor schema generation pipeline

A.1 Descriptor generation pipeline

We use the 70 billion parameter instruction fine-tuned version of Llama 3.3 to label our data. We

find that this model generates high-quality responses while still being practical to run on large amounts of data.

The pipeline that creates the descriptor schema is divided into three steps. First comes the *Generation* step, in which the initial descriptors are generated. This step consists of three stages. After this comes the *Disambiguation* step. During this step, most duplicate descriptors are merged into one or split into separate descriptors. Finally comes the *Synonym merging* step during which synonymous or near-synonymous descriptors are merged.

A.2 Generation step

The generation step is divided into three stages: initial generation, rewriting, and revision. In the initial generation stage, the model is given one document and is prompted to "[d]escribe the document in aspects including, but not limited to, style, tone, genre, topic, domain, language, quality, sentiment, etc." The model is asked to generate descriptors based on what it believes is essential for understanding, summarizing, or rewriting the document. The instructions are kept purposefully vague to not guide the model too much in a predetermined direction and to capture a broad range of descriptors. For each tag, the model is also prompted to generate a definition, which should "explain its meaning and relevance to the document in 2 to 3 sentences". The purpose of the definition is to provide additional context which is needed in later stages of the pipeline.

Since our aim is to build a general annotation schema, we do not want the descriptors to be overly specific. However in practice, we noted the model had a tendency to generate overly specific descriptors. For this reason, we offer the model the option to add more detailed descriptions into a list of "specifics". These are only used in the initial generation step but a discarded later.

A rewriting stage follows the initial generation. To evaluate how well the meaning of the document is captured in the descriptors, we give the descriptors and specifics generated in the previous stage back to the model and ask it to rewrite the document that they describe.

The rewriting stage is followed by a revision stage. After generating descriptors, specifics and a rewrite based on them, the model is asked to evaluate the quality of the rewrite and revise the descriptors accordingly. The model is prompted to compare the original document and its rewrite and to generate a revised set of descriptors that would better capture the meaning of the document.

The revision stage is followed by another rewrite stage, this time using the revised descriptors as input. This is followed by another revision stage and finally one more rewrite. Although this process could be repeated arbitrarily many times, we stop after three sets of descriptors and rewrites. To identify the set of descriptors that best captured the meaning of the original document, we embed the original document and each of its rewrites using the Stella-en-400M-v5 model StellaEncoder (2024), which is one of the best performing models in its size for English text clustering on the MTEB leaderboard (Muennighoff et al., 2023). The cosine similarity between each rewrite and the original is used as a measure of how much the rewrite resembles the original. The closer a rewrite is to the original document, the better the descriptors have summarized the document and thus, the better their quality.

After these steps, for each document, we only keep the descriptors that produced the best rewrite (measured by cosine similarity) and discard the rest. The specifics are also discarded at this stage.

A.3 Disambiguation step

In the disambiguation stage, we normalize all tags to lowercase and group together all definitions for a given tag. The vast majority of tags have only one definition, but some have more, with the most common ones such as "informal tone" having thousands of different definitions. The tags with more than one definition are provided to the LLM, which is prompted to either merge together synonymous or near-synonymous entries and provide one holistic descriptor or split the group into several descriptors, each with a locally unique tag and definition. To avoid overwhelming the model with extremely long inputs, large groups are split into subgroups of 20 definitions or fewer, each of which is disambiguated separately. As the model is not aware of decisions made in different instances, it can generate new, locally unique descriptors that are still duplicates globally. Therefore, this process is repeated several times until there are only a handful of duplicates left.

After 20 iterations, only four tags remain with more than one definition. The process is stopped here, since the model keeps inventing descriptors

that already exists elsewhere in the data and the rate of lowering the number of duplicates approaches zero.

A.4 Synonym merging step

After disambiguation, there are still many synonymous and near-synonymous descriptors in the set because in the disambiguation step only exact tag duplicates were grouped together. For instance, tags such as "formal tone" and "formal language" are still separate, despite being arguably synonymous in most contexts. In the synonym merging step, each descriptor is embedded with the Stella-en-400M-v5 model and mutually nearest neighbors are provided to the LLM. The model is prompted to either accept them as synonyms and choose one as the representative, while the other is discarded. The discarded descriptor is thus effectively merged into the other. If the model finds the given descriptors to be sufficiently dissimilar, they are kept as separate entries. After each iteration, new mutually nearest neighbors are identified and provided to the model. The process ends when no more merges are made.

The synonym merging step merges most of the remaining duplicate tags that survived the disambiguation step, but leaves just one tag with more than one definition. To ensure that each tag has exactly one definition, we do a forced merge, where the remaining duplicates are provided to the model, which chooses one as the representative descriptor.

B Prompts

This Appendix contains all the prompts used in the pipeline.

B.1 Initial prompt

System prompt:

```
##Instruction:
You will be given a document.
   Your task is to create a
   comprehensive list of
   descriptors - words or phrases
    that distill the meaning,
   tone, style, genre, topics,
   and other characteristics of
   the document. Do not focus
   solely on the topics; also
   include phrases that describe
   the tone and style.
The descriptors you generate
   should be useful for two
   purposes:
1. Rewriting the document in a
   way that is faithful to the
   original.
2. Classifying the document in a
   large variety of NLP
   classification tasks.
##Requirements:
1. Generate Descriptors:
- Describe the document in
   aspects including, but not
   limited to, style, tone, genre
   , topic, domain, language,
   quality, sentiment etc.
   Describe anything that you
   believe is essential for
   understanding, summarizing, or
    rewriting the document.
- If necessary, you can add
   minute details into the "
   specifics" list. This list can
    contain individual words,
   phrases and other details that
    do not necessarily describe
   the document but are important
    for rewriting it.
2. Descriptor Details:
- Descriptors can be single words
    or multi-word phrases. They
   should typically not be longer
    than 3 words.
- Each descriptor should describe
    only one aspect of the
   document. Multi-aspect
   descriptors should be split
   into separate descriptors.
- The list should be so extensive
    that, given only the list, a
   competent person should be
   able to rewrite the original
   document.
- Do not repeat descriptors and
   avoid near-duplicates.
3. Order:
- Order the descriptors in order
   of importance: the most
   important descriptors should
   be listed first.
- Importance is determined by how
```

```
    essential the descriptor is
   for reconstructing,
   understanding, or summarizing
   the document.
4. Descriptor Explanations:
- For each descriptor, explain
   its meaning and relevance to
   the document in 2 to 3
   sentences.
- The items in the specifics list
    do not need to be explained.
5. Format:
- The output must be formatted as
    a JSON object. Follow this
   formatting exactly.
- Never add any preamble or
   anything else outside the JSON
    object.
- Separate the descriptors and
   their explanations with a
   semicolon ';'. Do NOT include
   a semicolon anywhere else in
   the descriptor.
- Example: {"descriptors": ["<
   descriptor>; <explanation>",
   "<descriptor>; <explanation>",
    "<descriptor>; <explanation
   >", ...], "specifics": ["<
   specifics>", "<specifics>", "<
   specifics>",...]}
```

User prompt:

```
    {document}
```

B.2 Rewrite prompt

System prompt:

```
##Instruction:
You will be given a list of
   descriptors and specifics:
- Descriptors: Describe the
   document in aspects including,
    but not limited to, style,
   tone, genre, topic, domain,
   language, quality, sentiment,
   etc.
- Each descriptor is followed by
   an explanation of its meaning
   and relevance to the document.
- The descriptors are ordered in
   order of importance: the most
   important descriptors are
   listed first.
- Specifics: Minute details such
   as individual words and
   phrases, emphasis, structure,
   etc. to help in rewriting the
   document.
##Your Task:
Reconstruct the original document
    as accurately as possible
   using both the descriptors and
    specifics provided. Do not
   describe the document; instead
   , produce the document itself.
##Requirements:
1. Utilize All Descriptors:
- Incorporate all descriptors and
    specifics into your writing.
- Ensure that the content, tone,
   style, and structure align
   with the descriptors.
- The descriptors are ordered in
   order of importance: the most
   important descriptors are
   listed first.
2. Format:
- The output must be formatted as
    a JSON object. Follow this
   formatting exactly.
- Never add any preamble or
   anything else outside the JSON
    object.
- Example: {{"text": "<rewritten
   version of the document>"}}
```

User prompt:

```
*Descriptors*:
<start>{descriptors}<end>
*Specifics*:
<start>{specifics}<end>
```

B.3 Revision prompt

System prompt:

```
##Instruction:
You will be given:
1. Two Documents:
- Original Document: The initial
   text.
- Rewritten Document: An attempt
   to recreate the original
```

```
   document based solely on
   provided descriptors and
   specifics.
2. A List of Descriptors and
   Specifics (describing the
   original document):
- Descriptors: Describe the
   document in aspects including,
    but not limited to, style,
   tone, genre, topic, domain,
   language, quality, sentiment,
   etc.
- Each descriptor is followed by
   an explanation of its meaning
   and relevance to the document.
- The descriptors are ordered in
   order of importance: the most
   important descriptors are
   listed first.
- Specifics: Minute details such
   as individual words and
   phrases, emphasis, structure,
   etc. to help in rewriting the
   document.
##Your Task:
1. Critique the Rewritten
   Document:
- Analyze the rewritten document.
- Point out ways in which the
   rewrite differs from the
   original. Point out specific
   sentences or phrases that
   diverge meaningfully from the
   original, if applicable.
- Focus on discrepancies in
   content, tone, style,
   structure, and any missing or
   misrepresented elements.
2. Revise the Descriptors and
   Specifics:
- Adjust and revise the
   descriptors and specifics to
   help in creating a more
   faithful rewrite of the
   original document and for
   classifying it in a large
   variety of NLP classification
   tasks.
- Each descriptor should describe
    only one aspect of the
   document. Multi-aspect
   descriptors should be split
   into separate descriptors.
- The specifics list can contain
   individual words, phrases and
   other details that do not
   necessarily describe the
   document but are important for
    rewriting it.
- The revised lists should not
   contain any direct or indirect
    references or comparisons to
   the original lists. They
   should stand on their own.
- For each descriptor, explain
   its meaning and relevance to
   the document in 2 to 3
   sentences. The specifics do
   not need to be explained.
##Requirements:
1. Specificity: Provide detailed
   observations in your critique.
2. Clarity: Ensure the revised
   descriptors are clear and
   precise.
3. Completeness: Cover all
   aspects where the rewritten
   document deviates from the
   original.
4. Conciseness: Do not make the
   descriptors overly long. They
   should typically not be longer
    than 3 words. Do not repeat
   descriptors and avoid near-
   duplicates.
5. Order: Order the descriptors
   in order of importance: the
   most important descriptors
   should be listed first.
6. Format:
- The output must be formatted as
    a JSON object. Follow this
   formatting exactly.
- Never add any preamble or
   anything else outside the JSON
    object.
- Separate the descriptors and
   their explanations with a
   semicolon ';'. Do NOT include
   a semicolon anywhere else in
   the descriptor.
- Example: {"differences": "<
```

```
   differences between original
   and rewrite>",
"descriptors": ["<descriptor>; <
   explanation>", "<descriptor>;
   <explanation>", "<descriptor>;
    <explanation>", ...], "
   specifics": ["<specifics>", "<
   specifics>", "<specifics>",
   ...]}
```

User prompt:

```
{
*Original Document*:
<start>{original_document}<end>
*Rewritten Document*:
<start>{rewritten_document}<end>
*Descriptors*:
<start>{descriptors}<end>
*Specifics*:
<start>{specifics}<end>
}
```

B.4 Disambiguation prompt

System prompt:

```
You will be given a descriptor
   and its list of explainers.
   The descriptor describes a
   document and the explainers
   provide additional context to
   the descriptor.
Each explainer line is prefixed
   with its INDEX in square
   brackets, like: [0] text...
Your job is to merge/split
   explainers into semantically
   consistent groups *and return
   the INDICES of the original
   explainers you placed into
   each group*. Do not invent new
    indices, and use each index
   exactly once across all groups
   .
\#\# Instructions
1) Read the descriptor and
   explainers (each line is like:
    [<index>] <explainer text>).
2) Compare the explainers to each
    other and consider:
    - Do they point to the same
       kind of interpretation,
       rationale, or feature?
    - Do they reflect a
       consistent understanding
       or application of the
       descriptor?
3) Based on this comparison,
   create one or more
   semantically coherent groups.
4) For each group, provide:
   - group_descriptor (<= 3 words
       where possible. Coherent
      with the explainers in this
       group.)
   - group_explainer (Concise,
      canonical summary that
      covers the meaning of all
      explainers in this group)
   -
      original_explainer_indices_in_this_grou
      : an array of the *indices*
       you grouped here (e.g.,
      [0, 5, 7]).
Notes:
- Do NOT drop any original
   explainer indices. Every
   original explainer index must
   appear exactly once across
   groups.
- You may keep the original
   descriptor for one group;
   newly invented descriptors are
    allowed for other groups.
- Make minimal changes to the
   original explainers when
   forming group_explainers;
   focus on summarizing and
   unifying the meanings.
- Descriptors and explainers must
    be self-contained -- they
   must not contain references or
    comparisons to the original
   descriptors or explainers or
   the other groups.
\#\# Output format
Return a single valid JSON object
    with this exact structure (no
    extra text or commentary of
   any kind):
{"original_descriptor": "<
   original descriptor>",
```

```
  "groups": [
    {"group_descriptor": "<group
       descriptor>", "
       group_explainer": "<group
       explainer>", "
       original_explainer_indices_in_this_group
       ": [<index_a>, <index_b>,
       ...]}
  ]}
```

User prompt:

```
Descriptor: {descriptor}
Explainers (each prefixed by its
   INDEX):
{explainers}
```

B.5 Synonym merging prompt

System prompt:

```
You are deciding whether two
   descriptor-explainer pairs, A
   and B, refer to the same
   concept.
The descriptors describe a
   document, while the explainer
   provides additional context.

Treat them as synonyms (set
   is_synonym=true) if any of the
    following is true:
    - They refer to the same
       concept or entity despite
       different wording.
    - They deviate slightly in
       specificity but still
       cover the same core
       meaning.
    - They differ in phrasing but
        could be used
       interchangeably in most
       contexts.
    - They differ in word order,
       punctuation or spelling
       but mean the same thing.
    - They list similar things in
        different order.

Do NOT treat them as synonyms (
   set is_synonym=false) if any
   of the following is true:
    - They refer to different
       concepts/entities or
       mutually exclusive
       categories.
    - One is substantially
       broader/narrower in scope
       or specificity.
    - The explainer introduces a
       difference in meaning even
        when descriptors are
       similar.

In edge cases, prefer to NOT
   treat as synonyms.

If synonyms:
    - choose which to keep: "A"
       or "B" (prefer clearer,
       more general phrasing);
    - set keep to the chosen
       label and drop to the
       other.

If not synonyms:
    - Still return keep and drop
       as A and B (order doesn't
       matter).

Return ONLY JSON with keys:
   is_synonym (bool), keep ("A"|"
   B"), drop ("A"|"B").
Do not rewrite or invent
   descriptors.
```

User prompt:

```
A: descriptor: {descriptorA}
   explainer: {explainerA}
==============
B: descriptor: {descriptorB}
   explainer: {b.explainerB}
```

B.6 Force merging prompt

System prompt:

```
You are selecting a
   representative pair among
   candidates that share the same
    descriptor.
The descriptor describes a
   document, while the explainer
   provides additional context.
Your task is to choose the most
   representative pair among the
   given pairs.
```

```
Return the integer index of the
   pair to keep in the field '
   keep'
When choosing the representative
   pair, prefer concise, general,
    commonly used phrasing that
   covers the meaning of all
   pairs.
Return ONLY JSON with: keep (int)
   . Do not invent or rewrite
   anything.
```

User prompt:

```
index: {i} | descriptor: {
   descriptorA} | explainer: {
   explainerA}
index: {j} | descriptor: {
   descriptorA} | explainer: {
   explainerB}
...
```

B.7 Harmonization prompt

System prompt:

```
You are a careful ontology
   assistant.
Task: Given a query pair in the
   form 'descriptor; explainer'
   and a list of schema
   candidates, pick the single
   candidate that is synonymous
   or near-synonymous in most
   contexts to the query.
If many candidates are synonymous
   , choose the one that is the
   closest match.
If you find a suitable candidate,
    respond with is_synonym=true
   and set chosen_index to
   exactly the chosen candidate's
    index.
If none are acceptable, return
   is_synonym=false.
Output MUST be JSON only.
```

User prompt:

```
Query: {descriptor}; {explainer}
Candidates:
{i} {candidateA}
{j} {candidateB}
...
```

C Example document and descriptors

Document

- I am now the proud owner of a beautiful Black (the fastest colour of course) GCSRT8 here in Western Australia. There is a growing number of these beasts here although I have yet to see one on the road. You seem to have a fantastic SRT8 community thing going in this forum, I only wish that there was a GCSRT8 group like you here. (If there is an active SRT8 forum/group in Oz, can someone let me have the details?) I have gathered a lot of useful info from the forum to help me with my next move regarding performance mods and have decided the first will be to change my tstat to a 180 (and maybe the fan mod - although I am not that comfortable in cutting wires etc to get it done) Can anyone please advise where I can get these thermostats in Australia? What brands are available here? Part numbers? Our local auto chain store (Supa Cheap) hardly even knows what they are let alone if they stock them! Any help would be appreciated.

Raw descriptors

- car enthusiast; The document is written by a car enthusiast, indicating a strong interest in vehicles and their modification. The author's passion for cars is evident in their language and tone.
- technical; The document discusses technical aspects of car modification, such as thermostats and fan mods. This indicates a focus on the mechanical details of the vehicle.
- Australian; The document is set in Western Australia, and the author is seeking information specific to their location. This context is important for understanding the author's needs and concerns.
- informal; The document has a casual and friendly tone, indicating that the author is approachable and enthusiastic about the topic. The use of colloquial language and expressions conveys a sense of camaraderie.
- modification-focused; The document discusses plans for modifying the vehicle's performance, including changing the thermostat and potentially installing a fan mod. This

indicates a focus on improving the vehicle's capabilities.

- seeking advice; The author is seeking guidance and recommendations from the community, demonstrating a willingness to learn and take advice from others. This shows that the author values the opinions and expertise of others.

- personal; The author is a proud owner of a GC-SRT8 and is seeking to modify and improve their vehicle, indicating a personal investment in the topic. The author's ownership and enthusiasm are central to the document.

- regional; The author is seeking information specific to Australia, including availability of parts and local resources. This context is crucial for understanding the author's needs and concerns.

- appreciative; The author expresses gratitude for the community's potential help and advice, demonstrating a positive and respectful attitude. This tone is important for building relationships and trust within the community.

- enthusiastic; The author is excited about their vehicle and the potential modifications, conveying a sense of eagerness and anticipation.

- community-oriented; The author is seeking to connect with others who share their interests and is looking for a sense of community and belonging.

Harmonized descriptors

- automotive enthusiast; The author is an automotive enthusiast with a strong interest in modifying and improving vehicles, possessing deep knowledge of cars and trucks, and valuing high-quality service.

- automotive performance; The document focuses on the automotive domain, discussing vehicle performance, modifications, and technical aspects.

- western australia context; The text is specific to Western Australia, highlighting the regional focus of the document.

- informal social; The tone of the document is casual and friendly, indicating a personal or social relationship between the writer and the reader, and describes a situation that occurs in a social setting, making it interesting and engaging, and involves colleagues in a light-hearted activity.

- performance mods; The document discusses various modifications made to vehicles to improve their performance, including upgrades to engines, transmissions, suspensions, and other parts, as well as customization and DIY efforts.

- user seeking advice; This descriptor captures the purpose of the post, which is for the user to seek information or advice from others.

- passion and identity; The author's passion for cars is a central aspect of their identity and is reflected in their language and experiences, with a focus on modifying and customizing their vehicle.

- relevance to australia; The document is specific to the Australian context, making it relevant to an Australian audience, with references to local policies, laws, and regulations.

- polite and appreciative tone; The tone is polite and appreciative, with the authors expressing gratitude for any help they might receive.

- enthusiastic mood; The tone is positive and enthusiastic, conveying excitement and anticipation.

- online community seeker; The author is looking for a community where they can express themselves freely and engage in discussions with others who share similar interests. This descriptor emphasizes the author's desire for connection and dialogue.

D Query phrases

The exact phrases used to query documents. The phrasing of the FAQ and Legal Notices queries were adapted from the label definitions used in WebOrganizer.

- **FAQ**: "faq; The page content is in the Frequently Asked Questions format"

- **Legal Notices**: "legal notices; Contains for example terms of service, legal disclaimers, privacy policies or license agreements"
- **Sarcasm**: "sarcasm; the document contains sarcastic humor and irony"

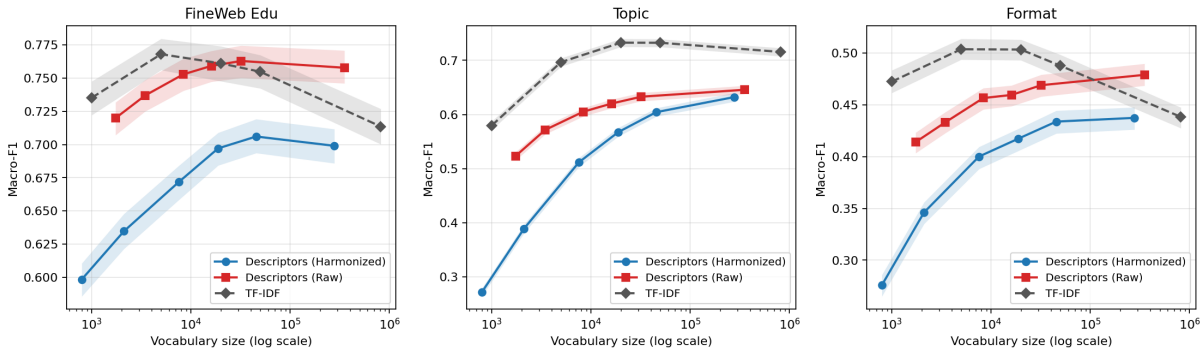

Figure 7: Macro-F1 as a function of feature-set size for harmonized descriptors, raw descriptors, and TF-IDF, across the three tasks. Descriptor vocabulary size is swept by minimum token frequency; TF-IDF by maximum vocabulary size. Shaded bands are 95% bootstrap confidence intervals.

E Classification results

Task	Features	Macro-F1 [95% CI]	W-F1	$\lvert V \rvert$
FineWeb Edu	Descriptors (Harmonized)	0.70 [0.69, 0.71]	0.92	278,093
FineWeb Edu	Descriptors (Raw)	0.76 [0.75, 0.77]	0.93	351,003
FineWeb Edu	TF-IDF	0.75 [0.74, 0.77]	0.94	50,000
Topic	Descriptors (Harmonized)	0.63 [0.62, 0.64]	0.64	278,093
Topic	Descriptors (Raw)	0.65 [0.64, 0.65]	0.66	351,003
Topic	TF-IDF	0.73 [0.72, 0.74]	0.75	50,000
Format	Descriptors (Harmonized)	0.44 [0.43, 0.45]	0.53	278,093
Format	Descriptors (Raw)	0.48 [0.47, 0.49]	0.57	351,003
Format	TF-IDF	0.49 [0.48, 0.50]	0.61	50,000

Table 5: Classification F1 comparing harmonized vs. raw descriptors, with TF-IDF as a text-based reference. Confidence intervals from 1000 bootstrap resamples of the test set. Descriptor configs use minimum frequency 1 (full vocabulary); TF-IDF capped at 50,000 features. $\lvert V \rvert$ = vocabulary size.

	Desc. (Harm.)			Desc. (Raw)			TF-IDF			
Class	Scan%	Lift	Prec.	Scan%	Lift	Prec.	Scan%	Lift	Prec.	N
edu	16.9%	4.7	37.9%	14.5%	5.5	44.0%	10.9%	7.4	58.7%	1,597
not-edu	74.4%	1.1	98.9%	74.3%	1.1	99.1%	73.8%	1.1	99.7%	18,403
Weighted mean	69.8%	1.4	94.0%	69.5%	1.4	94.7%	68.8%	1.6	96.5%	20,000

Table 6: Per-class retrieval at 80% recall, FineWeb Edu. Scan% = fraction of test set scanned to reach the target recall (lower better); Lift = precision/base rate; Prec. = precision at that point; N = support. Min. frequency 1; TF-IDF capped at 50,000. Sorted by harmonized Scan%.

Class	Desc. (Harm.)			Desc. (Raw)			TF-IDF			N
	Scan%	Lift	Prec.	Scan%	Lift	Prec.	Scan%	Lift	Prec.	
edu	2.0%	9.8	77.9%	2.0%	10.2	81.8%	1.7%	11.9	94.7%	1,597
not-edu	18.4%	1.1	99.9%	18.4%	1.1	100.0%	18.4%	1.1	100.0%	18,403
Weighted mean	17.1%	1.8	98.1%	17.1%	1.8	98.5%	17.1%	1.9	99.5%	20,000

Table 7: Per-class retrieval at 20% recall, FineWeb Edu. Columns as in Table 6.

Class	Desc. (Harm.)			Desc. (Raw)			TF-IDF			N
	Scan%	Lift	Prec.	Scan%	Lift	Prec.	Scan%	Lift	Prec.	
Adult	2.2%	35.7	26.1%	5.6%	14.2	10.4%	2.5%	32.3	23.6%	146
Religion	4.2%	18.9	39.2%	3.8%	21.3	44.0%	3.0%	26.3	54.4%	414
Travel	4.5%	17.8	42.3%	4.3%	18.5	43.9%	3.0%	26.7	63.2%	474
Software Dev.	4.5%	17.6	29.6%	4.9%	16.5	27.7%	2.9%	27.4	46.1%	336
History	7.4%	10.8	13.0%	4.6%	17.5	21.1%	3.4%	23.4	28.2%	241
Crime & Law	8.0%	10.0	35.9%	7.7%	10.4	37.2%	4.9%	16.4	59.1%	719
Hardware	8.3%	9.7	24.6%	8.1%	10.0	25.3%	3.9%	20.5	52.0%	508
Games	8.6%	9.3	28.2%	10.0%	8.0	24.3%	5.8%	13.9	42.4%	608
Fashion & Beauty	8.7%	9.2	34.9%	9.2%	8.7	33.0%	4.2%	18.9	71.9%	760
Science & Tech.	8.8%	9.1	36.9%	9.0%	8.9	36.1%	6.7%	12.0	48.4%	808
Food & Dining	9.2%	8.7	38.7%	10.0%	8.0	35.8%	4.5%	18.0	80.3%	894
Health	9.7%	8.3	50.7%	8.5%	9.4	57.4%	6.7%	11.9	73.0%	1,225
Software	11.0%	7.3	21.0%	9.5%	8.4	24.3%	5.8%	13.8	39.9%	579
Sports & Fitness	11.2%	7.1	56.8%	10.6%	7.5	59.9%	7.7%	10.4	83.1%	1,595
Politics	11.3%	7.1	46.2%	11.1%	7.2	47.1%	8.5%	9.4	61.3%	1,305
Entertainment	11.8%	6.8	50.6%	14.0%	5.7	42.7%	8.4%	9.5	70.9%	1,497
Education & Jobs	12.8%	6.3	37.0%	12.9%	6.2	36.7%	7.6%	10.6	62.3%	1,180
Industrial	13.3%	6.0	17.5%	11.3%	7.1	20.6%	6.1%	13.1	37.7%	578
Literature	13.7%	5.8	22.4%	14.2%	5.6	21.6%	8.3%	9.6	36.8%	766
Art & Design	14.4%	5.5	17.2%	10.5%	7.6	23.6%	5.0%	15.9	49.4%	620
Finance & Business	16.0%	5.0	37.5%	14.7%	5.5	41.0%	10.4%	7.7	57.7%	1,503
Home & Hobbies	16.7%	4.8	32.9%	17.4%	4.6	31.6%	10.0%	8.0	54.9%	1,374
Transportation	16.8%	4.8	16.0%	18.2%	4.4	14.8%	5.0%	15.8	53.2%	672
Social Life	18.7%	4.3	25.7%	17.0%	4.7	28.2%	12.7%	6.3	37.8%	1,198
Weighted mean	11.8%	7.8	36.3%	11.6%	7.9	36.8%	7.2%	12.9	58.4%	20,000

Table 8: Per-class retrieval at 80% recall, Topic. Columns as in Table 6.

Class	Desc. (Harm.) Scan%	Lift	Prec.	Desc. (Raw) Scan%	Lift	Prec.	TF-IDF Scan%	Lift	Prec.	N
Adult	0.1%	132.6	96.8%	0.2%	128.4	93.8%	0.2%	128.4	93.8%	146
History	0.3%	63.5	76.6%	0.3%	58.1	70.0%	0.3%	65.6	79.0%	241
Software Dev.	0.5%	44.0	73.9%	0.5%	40.9	68.7%	0.4%	46.0	77.3%	336
Religion	0.5%	43.1	89.2%	0.5%	39.7	82.2%	0.5%	42.7	88.3%	414
Travel	0.6%	36.8	87.2%	0.6%	33.4	79.2%	0.5%	39.3	93.1%	474
Hardware	0.6%	33.7	85.7%	0.7%	28.9	73.4%	0.6%	32.6	82.9%	508
Games	0.6%	31.4	95.3%	0.7%	29.5	89.7%	0.6%	32.4	98.4%	608
Art & Design	0.8%	25.8	80.0%	0.9%	22.9	70.9%	0.7%	28.6	88.6%	620
Crime & Law	0.8%	25.2	90.6%	0.8%	24.3	87.3%	0.8%	26.5	95.4%	719
Fashion & Beauty	0.8%	24.1	91.6%	0.8%	23.5	89.4%	0.8%	26.0	98.7%	760
Transportation	0.9%	22.8	76.7%	1.0%	20.0	67.2%	0.8%	26.1	87.7%	672
Food & Dining	0.9%	21.8	97.3%	0.9%	21.8	97.3%	0.9%	21.9	97.8%	894
Science & Tech.	0.9%	21.6	87.1%	1.1%	18.7	75.3%	0.9%	21.4	86.6%	808
Software	1.0%	20.9	60.4%	1.0%	19.8	57.4%	0.8%	24.6	71.2%	579
Industrial	1.0%	20.6	59.5%	1.0%	19.5	56.3%	0.7%	26.9	77.8%	578
Literature	1.3%	15.3	58.8%	1.3%	15.5	59.5%	0.9%	22.8	87.5%	766
Health	1.4%	14.7	90.1%	1.4%	14.7	90.1%	1.3%	15.3	93.5%	1,225
Education & Jobs	1.4%	14.5	85.8%	1.4%	14.1	83.1%	1.3%	15.5	91.5%	1,180
Politics	1.6%	12.7	82.9%	1.6%	12.1	78.8%	1.4%	14.2	92.9%	1,305
Sports & Fitness	1.6%	12.5	100.0%	1.6%	12.3	98.5%	1.6%	12.4	99.1%	1,595
Entertainment	1.6%	12.3	92.3%	1.7%	12.0	89.8%	1.6%	12.9	96.8%	1,497
Home & Hobbies	1.7%	11.8	81.4%	1.8%	11.3	77.7%	1.5%	13.6	93.5%	1,374
Finance & Business	1.7%	11.6	87.0%	1.9%	10.9	81.6%	1.6%	12.4	93.5%	1,503
Social Life	2.0%	9.8	58.8%	1.9%	10.4	62.5%	1.4%	13.9	83.0%	1,198
Weighted mean	1.3%	19.8	83.8%	1.3%	18.8	80.1%	1.1%	21.5	91.2%	20,000

Table 9: Per-class retrieval at 20% recall, Topic. Columns as in Table 6.

Class	Desc. (Harm.) Scan%	Lift	Prec.	Desc. (Raw) Scan%	Lift	Prec.	TF-IDF Scan%	Lift	Prec.	N
Legal Notices	1.5%	55.4	34.4%	1.2%	69.2	42.9%	1.2%	65.0	40.3%	124
Documentation	4.6%	17.5	8.9%	3.5%	22.8	11.6%	1.4%	58.0	29.6%	102
Academic Writing	5.0%	16.2	25.9%	3.5%	22.6	36.1%	2.9%	27.6	44.1%	319
Creative Writing	5.9%	13.6	16.4%	5.3%	15.2	18.4%	7.1%	11.2	13.6%	242
User Review	9.8%	8.1	15.2%	7.9%	10.1	18.8%	7.7%	10.4	19.3%	372
Comment Section	12.3%	6.5	24.5%	12.1%	6.6	24.9%	8.1%	9.9	37.0%	751
About (Pers.)	13.0%	6.2	11.4%	10.1%	7.9	14.6%	5.3%	15.2	28.1%	369
Tutorial	14.6%	5.5	25.2%	11.0%	7.3	33.4%	8.3%	9.6	44.3%	920
Customer Support	15.7%	5.1	8.1%	12.8%	6.3	9.9%	7.2%	11.2	17.8%	318
About (Org.)	16.1%	5.0	17.5%	14.9%	5.4	18.9%	9.5%	8.4	29.7%	704
Spam / Ads	20.5%	3.9	9.3%	14.9%	5.4	12.7%	4.9%	16.4	38.8%	473
Q&A Forum	22.5%	3.6	2.7%	14.0%	5.8	4.4%	6.0%	13.4	10.2%	153
News Article	23.2%	3.5	53.3%	22.0%	3.6	56.1%	16.7%	4.8	74.3%	3,092
News (Org.)	23.7%	3.4	20.8%	20.6%	3.9	23.8%	16.5%	4.9	29.8%	1,227
Personal Blog	23.7%	3.4	50.2%	25.9%	3.1	45.9%	24.0%	3.3	49.5%	2,974
Knowledge Article	24.4%	3.3	14.4%	20.5%	3.9	17.1%	11.8%	6.8	29.6%	877
Product Page	25.1%	3.2	54.9%	26.1%	3.1	52.9%	21.4%	3.7	64.4%	3,444
Structured Data	25.3%	3.2	5.1%	24.6%	3.2	5.2%	9.5%	8.4	13.4%	320
Audio Transcript	28.1%	2.8	4.3%	23.6%	3.4	5.1%	13.4%	6.0	8.9%	300
Nonfiction Writing	31.0%	2.6	14.9%	29.5%	2.7	15.6%	19.7%	4.1	23.4%	1,152
Listicle	34.7%	2.3	5.7%	25.9%	3.1	7.6%	19.0%	4.2	10.4%	492
Truncated	47.6%	1.7	3.1%	41.7%	1.9	3.6%	13.5%	5.9	11.0%	371
FAQ	49.2%	1.6	0.7%	29.9%	2.7	1.1%	13.4%	6.1	2.5%	84
Content Listing	52.5%	1.5	6.2%	44.4%	1.8	7.4%	24.4%	3.3	13.4%	820
Weighted mean	23.9%	4.3	33.0%	22.3%	4.9	33.8%	16.3%	6.8	43.7%	20,000

Table 10: Per-class retrieval at 80% recall, Format. Columns as in Table 6.

Class	Desc. (Harm.)			Desc. (Raw)			TF-IDF			N
	Scan%	Lift	Prec.	Scan%	Lift	Prec.	Scan%	Lift	Prec.	
Legal Notices	0.1%	161.3	100.0%	0.1%	155.1	96.2%	0.1%	155.1	96.2%	124
Documentation	0.3%	79.2	40.4%	0.1%	152.5	77.8%	0.2%	117.6	60.0%	102
FAQ	0.3%	68.6	28.8%	1.2%	17.5	7.4%	0.2%	88.0	37.0%	84
Audio Transcript	0.4%	51.9	77.9%	0.6%	34.5	51.7%	0.4%	53.3	80.0%	300
Creative Writing	0.4%	47.6	57.6%	0.5%	43.5	52.7%	0.3%	75.0	90.7%	242
Academic Writing	0.5%	45.1	71.9%	0.4%	50.8	81.0%	0.4%	51.4	82.0%	319
Q&A Forum	0.6%	36.5	27.9%	0.7%	27.2	20.8%	0.3%	63.3	48.4%	153
User Review	0.6%	31.8	59.1%	0.7%	28.2	52.4%	0.6%	32.5	60.5%	372
About (Pers.)	0.7%	31.1	57.4%	0.6%	33.1	61.2%	0.5%	37.8	69.8%	369
Spam / Ads	0.7%	29.3	69.3%	0.8%	24.8	58.6%	0.5%	37.9	89.6%	473
Customer Support	0.8%	24.8	39.5%	0.8%	25.0	39.8%	0.5%	38.3	61.0%	318
Structured Data	0.8%	24.4	39.0%	0.8%	23.4	37.4%	0.4%	48.2	77.1%	320
Comment Section	1.0%	19.8	74.4%	1.0%	19.9	74.8%	0.8%	23.7	88.8%	751
Tutorial	1.1%	18.0	82.9%	1.1%	17.8	81.8%	1.1%	18.3	84.4%	920
About (Org.)	1.4%	14.8	52.0%	1.4%	14.3	50.4%	1.0%	19.4	68.5%	704
Knowledge Article	1.6%	12.4	54.3%	1.7%	11.9	52.4%	1.2%	16.4	72.1%	877
News (Org.)	1.8%	10.9	66.8%	2.0%	9.8	60.0%	1.7%	11.9	73.0%	1,227
Listicle	2.0%	10.0	24.5%	2.1%	9.6	23.7%	1.2%	16.2	39.8%	492
Content Listing	2.7%	7.5	30.8%	3.4%	5.8	23.8%	1.8%	11.1	45.4%	820
Truncated	3.0%	6.8	12.7%	2.5%	8.0	14.8%	0.5%	37.4	69.4%	371
Nonfiction Writing	3.1%	6.6	37.8%	2.9%	6.9	39.6%	2.1%	9.7	55.8%	1,152
News Article	3.6%	5.6	86.6%	3.8%	5.3	82.5%	3.2%	6.2	95.5%	3,092
Personal Blog	3.6%	5.5	82.3%	3.7%	5.4	80.5%	3.4%	5.9	88.1%	2,974
Product Page	3.8%	5.2	89.8%	4.0%	5.1	87.0%	3.8%	5.3	91.9%	3,444
Weighted mean	2.6%	13.6	69.5%	2.7%	13.1	66.8%	2.2%	17.3	80.1%	20,000

Table 11: Per-class retrieval at 20% recall, Format. Columns as in Table 6.

Class	Desc. (Harm.)	Desc. (Raw)	TF-IDF
edu	general context (35), development narratives (18), scientific-research (308), informing the reader (37), use of etymology (19), word history (26), holiday history (8), anti-bullying strategy (9), discovery and artifact context (7), teacher reflection (11)	scientific topic (53), educational resource (94), educational topic (45), historical account (212), scientific language (279), environmental topic (59), educational content (695), historical event (84), historical narrative (119), scientific study (143)	or (46,836), of the (53,488), species (1,197), century (2,786), of (75,396), scientists (1,195), researchers (1,416), nasa (345), used (14,612), children (6,776)
not-edu	product description (1,028), international travel guide (237), customer centricity (504), sports article (489), local news update (239), opinionated writing (196), local news (612), professional language use (762), tone - promotional (2,231), personal story (241)	home improvement (284), sports news (1,681), product description (2,078), varied topics (177), travel guide (473), financial (232), professional language (1,292), business news (493), respectful dialogue (92), customer-centric (1,279)	my (21,750), me (16,737), 2018 (2,622), 2014 (3,109), our (29,795), we (38,363), business (10,558), 2019 (2,062), am (10,094), 2016 (3,171)

Table 12: Top-10 features per class for FineWeb Edu, ranked by logistic regression coefficient, comparing harmonized descriptors, raw descriptors, and TF-IDF. Parenthesised numbers are training-set document frequencies. Descriptor tags shown; definitions omitted. Descriptors at min. frequency 1; TF-IDF capped at 50,000.