Context-Aware Search Space Adaptation of Hyperparameters and Architectures for AutoML in Text Classification

Parisa Safikhani[1,2,*], David Broneske[1]

[1]The German Centre for Higher Education Research and Science Studies (DZHW), Germany
[2]Otto von Guericke University, Germany

Abstract

While Automated Machine Learning (AutoML) systems have shown strong performance on structured data, their application to natural language processing (NLP) tasks remains limited by static, task-agnostic search spaces. In this work, we propose a context-aware extension of AutoPyTorch that dynamically adapts both the hyperparameter search space and neural architecture configuration based on corpus-level meta-features. Our approach extracts interpretable textual statistics—such as average sequence length, vocabulary richness, and class imbalance—to guide the configuration of key hyperparameters. We also introduce two adaptive neural backbones, whose structures are shaped by these meta-features to improve model expressiveness and generalization. Experiments on 20 diverse text classification datasets—including subsets of GLUE, selected Kaggle benchmarks, and private corpora—demonstrate consistent performance improvements over strong baselines, particularly on datasets with limited training samples or severe class imbalance. Our results highlight the effectiveness of integrating dataset-level insights into the AutoML search process for NLP.

Keywords

AutoML, Text Classification, Hyperparameter Optimization, Meta-Features, Neural Architecture Search, Context-Aware Modeling, AutoPyTorch

1. Introduction

AutoML frameworks have significantly advanced the democratization of machine learning by automating the design and optimization of learning pipelines. While these systems have shown strong performance on structured data, their extension to NLP tasks remains limited due to the inherent complexity and diversity of textual data. Text classification, a core NLP task, presents unique challenges stemming from variable input lengths, diverse syntactic structures, and high lexical variation—factors that are often overlooked in conventional AutoML workflows.

Most current AutoML approaches for NLP adopt static pipeline configurations and search spaces, treating all datasets uniformly regardless of their linguistic characteristics. Even when modern frameworks include neural networks or transformer models, their hyperparameter search is usually performed within generic, manually designed boundaries. This static design neglects crucial dataset-specific properties such as text length distribution, vocabulary richness, or class imbalance, which are known to affect both model architecture performance and training dynamics [1, 2]. As a result, these frameworks may perform poorly on unusual or domain-specific text datasets, where generic configurations fail to address context-specific requirements.

To address this gap, we propose a context-aware extension of an AutoML Framework that dynamically adapts its hyperparameter search space and model architecture decisions based on corpus-level meta-features. Our approach integrates a systematic extraction of statistical and linguistic characteristics from each dataset—such as text length variability, lexical diversity, sample size, and class distribution—and uses these to inform both the configuration of search spaces and the structural design of neural backbones. By leveraging these insights, the system can better align model complexity, optimization schedules, and architectural choices with the demands of the data.

This paper makes two main contributions: First, we introduce a context-aware mechanism for dynamically adapting the hyperparameter search space in AutoML based on text-level meta-features such as text length, vocabulary diversity, and class imbalance. This enables the AutoML process to tailor its optimization bounds—e.g., for batch size, learning rate, and dropout—according to the statistical profile of each dataset. Second, we propose two adaptive neural backbones, *MetaMLP* and *ContextualAttentionNet*, whose configurations are shaped by statistical and lexical characteristics of the input text. These

CLiC-it 2025: Eleventh Italian Conference on Computational Linguistics, September 24 — 26, 2025, Cagliari, Italy
*Corresponding author.
safikhani@dzhw.eu (P. Safikhani); broneske@dzhw.eu (D. Broneske)
0000-0003-3029-7581 (P. Safikhani); 0000-0002-9580-740X (D. Broneske)

backbones enable the system to construct models from scratch that better reflect the structural and distributional properties of the data. Together, these innovations facilitate a more robust and efficient adaptation of AutoML pipelines to the unique demands of text classification tasks.

2. Related Works

Automated Machine Learning (AutoML) aims to streamline model development by automating the processes of feature engineering, model selection, and hyperparameter tuning. While AutoML has become widely successful for structured data, its adaptation to natural language processing (NLP) tasks, particularly text classification, poses unique challenges due to the complexity and diversity of textual data. In this section, we review existing research relevant to AutoML applications in NLP, focusing on the limitations of current AutoML frameworks, the role of dataset-driven meta-features, and recent developments in customizing both hyperparameter search spaces and neural architectures specifically tailored to text data.

2.1. AutoML for NLP Tasks and Search Space Design

Automated machine learning (AutoML) has traditionally excelled on structured (tabular) data, whereas applying it to raw text required additional effort to convert text into features [2]. In recent years, several AutoML frameworks have been extended to handle text classification, integrating NLP-specific models and pipeline steps. For example, AutoGluon and AutoKeras can handle deep NLP models (including modern transformers) for classification, with search spaces that encompass state-of-the-art architectures like BERT and RoBERTa [3, 4]. AutoKeras even adjusts its search space based on the task modality: it detects when the input is text and accordingly includes appropriate text vectorization and neural network blocks in the configuration space [3]. Cloud-based AutoML services such as Azure AutoML typically treat text as generic input features (e.g., via TF-IDF or bag-of-words) and do not customize hyperparameter settings based on dataset-specific characteristics [5].

Notably, researchers have evaluated general AutoML tools on NLP tasks by converting text to fixed embeddings (e.g. using Sentence-BERT to obtain features) to fit into a tabular AutoML pipeline [6, 7, 8]. These tools can discover effective models for text data, but they typically operate within broad, fixed search spaces and often lack mechanisms for fine-grained hyperparameter tuning tailored to a specific corpus. In other words, current AutoML frameworks for NLP tend to follow a one-size-fits-all approach, leaving potential efficiency gains from dataset-specific adaptation largely untapped. Moreover, they rely on selecting from existing machine learning or neural network architectures, rather than dynamically constructing models based on the unique characteristics of the textual data.

2.2. Meta-Features and Meta-Learning for AutoML in NLP

To guide model selection and hyperparameter optimization, many studies have leveraged dataset characteristics (so-called meta-features). Early work by Lam and Lai [1] characterized text datasets with a small set of features (e.g., number of documents, vocabulary size, average document length) to predict the classification error of different algorithms, thus recommending the best classifier for the task. This pioneering meta-learning approach demonstrated that simple corpus-level metrics can inform algorithm selection. Subsequent research greatly expanded the repertoire of meta-features for NLP. For instance, Madrid et al. [2] define 72 corpus-level attributes – covering general dataset properties, class imbalance, lexical diversity, stylometry, statistical measures, and readability indices – to drive automated selection of text representation techniques. Many of these features capture precisely the kind of information used in our approach for another goal, which is automatic customization of the search space and not a text representation method, such as average and standard deviation of document length, vocabulary richness (unique word ratios), number of classes, and so on. By extracting such meta-features from a new text dataset, one can compare them to previously seen tasks and infer which models or hyperparameter settings might be appropriate. Researchers have applied meta-features in various meta-learning systems for NLP. Gomez et al. [9] introduced an evolutionary meta-learning method (ELMR) that uses 11 statistical meta-features of a text corpus to evolve rules for selecting the optimal classifier. Their approach automatically learned decision rules (via a genetic algorithm) to identify, for example, when a Naïve Bayes vs. SVM vs. neural model would be most effective, based on corpus characteristics. In a broader approach, Ferreira and Brazdil [10] leveraged an active testing strategy to recommend full text-classification pipelines, evaluating candidate preprocessing methods and classifiers on small data samples and using meta-features to pick the best pipeline. Meta-learning has also been used to warm-start hyperparameter optimization in general AutoML frameworks. Ferreira and Brazdil [10] successfully employed 46 dataset descriptors to initialize Bayesian hyperparameter search in Auto-Sklearn, improving efficiency by starting from configurations that worked well on similar prior datasets. More recently, Desai et al. [11] built a text AutoML system that uses a minimal set of only three meta-features

(e.g. dataset size, average sentence length) to choose among three Transformer architectures (BERT, ALBERT, XLNet) for a classification task. Despite its limited scope (restricted to only a few models), this work demonstrated the promise of corpus features in guiding model selection for NLP. Our approach extends these concepts further by integrating a set of corpus-level characteristics to dynamically guide not only architecture selection but also hyperparameter tuning within AutoPyTorch, leveraging its capability to construct neural networks from scratch, which is essential for effectively handling the diverse and complex nature of textual data.

In summary, prior research shows that incorporating dataset-derived features, ranging from simple counts to complex linguistic metrics, can significantly enhance automated model selection and configuration in NLP. However, these approaches predominantly focus on selecting among predefined models or representations. To the best of our knowledge, this work is the first to dynamically adjust the hyperparameter search space itself based on dataset-derived meta-knowledge, specifically aimed at constructing deep learning models from scratch.

2.3. Hyperparameter Search Space Adaptation in AutoML

Typical AutoML systems rely on a fixed, expert-designed search space intended to be generic across many datasets. For example, Auto-WEKA formalized the Combined Algorithm Selection and Hyperparameter optimization (CASH) problem—searching over a joint space of 27 base classifiers, their respective hyperparameters, and various feature-selection techniques—using Bayesian optimization to navigate hundreds of parameters without dataset-specific specialization [12]. Auto-Sklearn similarly constructs a broad configuration space of 15 classifier types and over 110 hyperparameters (spanning preprocessing and classifiers) yet remains agnostic to the particular characteristics of the input data [13]. While such comprehensive spaces can cover many scenarios, they are often inefficient: many configurations may be irrelevant or suboptimal for a particular text dataset. For instance, a small set of short tweets likely does not require deep ensembles or large n-gram ranges, yet a static search space devotes trials indiscriminately to these options. This inefficiency has motivated research into reducing or tuning the search space based on prior knowledge.

One line of work is search space transfer via meta-learning. Wistuba et al. [14] first proposed to leverage experience from previous hyperparameter optimizations to constrain the search for a new task. In their approach, the hyperparameter space is narrowed to a region (defined by a center point and radius) believed to contain good solutions, effectively pruning away less promising regions. They explored designing a smaller, task-specific search space for the target problem instead of using the default full space. By transferring knowledge of what configurations worked well on similar datasets, these methods aim to accelerate HPO by focusing on the most relevant parts of the space. Such techniques have shown benefits in general AutoML settings, reducing the dimensionality or bounds of hyperparameters to improve search efficiency. However, applying this idea in the NLP domain remains relatively unexplored – current AutoML tools do not automatically adjust fundamental hyperparameter ranges (e.g. maximum vocabulary size, network depth, learning rate schedules) based on text-specific data characteristics. The search space is usually defined a priori (often by human experts) and stays fixed regardless of whether the text data consists of tweets or pages of encyclopedia, or whether the vocabulary is 500 words or 50,000 words.

Recently, a few nascent approaches have hinted at the potential of dataset-driven search space adaptation. Notably, Zero-Shot AutoML techniques combine meta-learning with model selection to configure pipelines without any trial-and-error on the new data. For example, the ZAP framework by Öztürk et al. [15] attempts to directly select a pretrained model and its fine-tuning hyperparameters for a new dataset in a zero-shot manner. ZAP trains a meta-model on a large collection of prior tasks, using only trivial meta-features of each dataset (such as image resolution or the number of classes) to predict the best pipeline. In their vision domain experiments, this approach could successfully pick an appropriate model and hyperparameter configuration without searching, underscoring that even coarse dataset descriptors can be informative for hyperparameter decisions. This idea is very much in line with our goal of text-aware search space customization. However, aside from such cutting-edge research prototypes, mainstream AutoML for NLP still lacks the capability to dynamically tailor the hyperparameter search space based on the dataset.

2.4. Text-Oriented Architecture Search & Pruning

Recent research in AutoML for NLP has focused on tailoring neural architectures to the needs of text data. Neural Architecture Search (NAS) techniques, when specialized for textual tasks, have proven effective in discovering model structures that outperform generic designs. For example, Wang et al. [16] propose TextNAS, a search space explicitly designed for text representation, and show that automatically discovered architectures can surpass manually crafted networks on sentiment analysis and inference tasks. These results highlight that text-specific search spaces – incorporating layers like CNNs or RNNs suited to sequence data – can yield state-of-the-art performance where off-the-shelf image-inspired architectures falter. In parallel, the emergence of large pre-trained language

models has motivated architecture pruning and adaptation strategies. Rather than treat one model size as fit-for-all, researchers leverage NAS to compress or select architectures appropriate for a given task's resource constraints. For instance, NAS-BERT uses neural architecture search to automatically prune BERT, producing a family of smaller models that retain accuracy across tasks while meeting various latency or memory requirements [17]. Collectively, these efforts underscore that architecture-level customization is crucial for optimizing NLP pipelines. By adjusting neural backbones to text characteristics (lengthy inputs, specialty domains, etc.), NAS and pruning approaches lay the foundation for more adaptive AutoML solutions.

While significant advances have been made in both meta-feature-driven hyperparameter tuning and architecture-level customization (NAS/pruning), these areas have evolved somewhat separately. To date, there remains an absence of integrated methods that dynamically combine architecture selection with hyperparameter optimization based on explicit text dataset characteristics. Our paper directly addresses this gap by introducing a unified framework within AutoPyTorch that adapts both model architectures and hyperparameter configurations based on corpus-specific meta-features. This approach ensures that every component of the AutoML pipeline—from model structure to training parameters—is tailored specifically for the dataset at hand, leading to a more efficient and robust text-classification solution.

3. Methodology

Our objective is to enhance the adaptability and performance of AutoML systems for text classification by dynamically customizing both the hyperparameter search space and neural architectures based on intrinsic properties of the input dataset. We implement this within the AutoPyTorch framework, which offers modular extensibility, fine-grained pipeline control, and full support for deep learning models constructed from scratch. This flexibility is especially valuable for textual data, where architectural decisions—such as incorporating attention mechanisms or shaping MLPs—must align with dataset-specific traits like sequence length, lexical diversity, and class imbalance [18, 19].

Unlike other AutoML frameworks that rely on fixed pipelines or pre-trained models, our approach enables the construction of neural architectures that are directly informed by corpus-level characteristics. Prior work has shown that such dynamic, data-driven architecture generation leads to better generalization and improved performance, particularly in heterogeneous or domain-specific scenarios [20, 21]. These findings motivate our design of a context-aware adaptation mechanism that leverages meta-features to steer both model configuration and training strategy during the AutoML search, effectively bridging the gap between static AutoML systems and the flexible demands of NLP tasks.

3.1. Text-Level Meta-Feature Extraction

To support both hyperparameter configuration and model architecture design (e.g., number of neurons and layers), we extract a comprehensive set of text-level meta-features using an enhanced analysis function. These include:

3.1.1. Text Length

Text length is a critical meta-feature in NLP that impacts both architecture selection and hyperparameter configuration. Short texts (e.g., fewer than 10 tokens) lack sufficient semantic context, leading to poor model performance, as shown in McCartney et al. [22]. Conversely, very long texts exceed transformer input limits (e.g., 512 tokens in BERT) and require either truncation or specialized architectures such as Hierarchical Attention Networks (HAN) [23] or Longformer [24].

To address these issues, we compute the average and standard deviation of text length at the corpus level and incorporate them into multiple stages of the AutoML pipeline. Specifically, long average sequence lengths trigger smaller batch sizes (e.g., 8–16 for texts >300 characters), shorter warm-up periods in cosine annealing schedules, and reduced learning rates to stabilize training. Additionally, we adapt the architectural shape of candidate MLP backbones: datasets with long inputs receive "long funnel" configurations to compress high-dimensional sequences, while very short texts invoke compact "diamond" shapes to avoid overfitting. High variance in length distribution increases regularization (via dropout) to ensure generalization across variable-length inputs.

This integration ensures that the AutoML system dynamically aligns model complexity and optimization behavior with the distributional characteristics of the input text, improving both efficiency and robustness in the search process.

3.1.2. Vocabulary Richness and Lexical Diversity

Vocabulary richness—commonly measured using the type-token ratio (TTR) or corpus-level approximations such as the unique-to-total word ratio—reflects the semantic complexity of a text corpus. Higher lexical diversity increases the dimensionality of the input space and often correlates with more complex linguistic structure [25, 26], requiring models with greater expressive capacity. From a theoretical standpoint, diverse corpora

demand models with higher VC dimension and wider hypothesis classes to capture nuanced patterns [27].

To account for this, our system dynamically adapts architectural complexity based on measured lexical diversity. For datasets with a high unique word ratio (e.g., > 0.3), we increase the number of neuron groups and expand the maximum layer width (`max_units`) in our text-aware MLP-based backbone, allowing the model to better capture semantic variation. Conversely, for low-diversity corpora, we reduce network width and depth to prevent overfitting. In addition, the backbone shape is adjusted: high-diversity texts favor "long funnel" architectures, while simpler datasets default to "diamond"-shaped or regular "brick-like" architectures composed of repeated modules. We also modify activation functions: when diversity is low and the default ReLU may underperform, GELU is automatically selected to improve representation power for simple patterns.

These adaptations ensure that both the search space and the resulting architectures reflect the semantic variability of the input corpus, allowing the AutoML process to match model expressiveness with linguistic richness.

3.1.3. Number of Samples

The number of training examples is a fundamental meta-feature that influences model complexity, training dynamics, and generalization behavior. Small datasets tend to increase the risk of overfitting—particularly when using high-capacity neural networks—whereas large datasets enable the use of deeper models, longer training schedules, and reduced regularization. This is grounded in statistical learning theory, which links generalization error to both the size of the hypothesis class and the number of available training samples [28]. Empirical studies support this connection: Domhan et al. [29] and Probst et al. [30] show that both training regimes and optimal hyperparameter values (e.g., learning rate, dropout) scale with dataset size.

In our approach, we compute the number of training samples during meta-feature extraction and use this to adapt the AutoML search space. For datasets with fewer than 1,000 examples, we expand the dropout search space (up to 0.8), reduce learning rates, and favor simpler backbones such as narrow MLPs or shallow attention blocks. Training budgets are also capped to avoid overfitting under data scarcity. In contrast, datasets with more than 10,000 samples prompt relaxed regularization and enable higher-capacity configurations, such as increased `max_units` and longer training horizons. These modifications ensure that the resulting models are appropriately scaled to the statistical regime of the dataset, improving both robustness and computational efficiency.

3.1.4. Label Distribution and Class Imbalance

Imbalanced class distributions are a common challenge in text classification, where certain categories (e.g., hate speech, fraud cases) are underrepresented but critically important. When the class imbalance ratio—the proportion between the most and least frequent class—exceeds a certain threshold, classification performance for minority classes deteriorates due to model bias toward majority labels [31, 32]. This bias arises from the difficulty of estimating rare class probabilities under skewed priors, which leads to inaccurate posterior approximations, especially when using symmetric loss functions such as cross-entropy.

In our AutoML framework, class imbalance is measured during the meta-feature extraction phase and directly influences the search space configuration. For datasets with imbalance ratios exceeding 3.0, we expand the dropout range (e.g., up to 0.8), reduce learning rates, and increase the warm-up period in cosine learning rate schedules. These measures are designed to stabilize training under uneven gradient updates and reduce overfitting to dominant classes. Conversely, for nearly balanced datasets (imbalance ratio below 1.5), regularization is relaxed to allow more expressive learning.

Although architectural constraints are not enforced strictly based on imbalance, our search space prioritizes configurations that are empirically robust to imbalance, such as residual-normalized attention layers or funnel-shaped MLPs. Together, these mechanisms enable the AutoML system to maintain balanced performance across both major and minor classes.

4. Experiments

To evaluate the effectiveness of our context-aware AutoML framework for text classification, we conducted comprehensive experiments on 20 diverse datasets. Our experiments were designed to compare the performance of our proposed context-aware AutoPyTorch against a strong baseline using static configurations in AutoPyTorch.

4.1. Datasets

We conduct experiments on a diverse collection of datasets, including a stratified 30% subset of each task from the GLUE benchmark [33], a widely used evaluation suite for natural language understanding. GLUE (General Language Understanding Evaluation) consists of multiple sentence-level and sentence-pair classification tasks, covering linguistic phenomena such as entailment, paraphrase detection, sentiment analysis, and grammaticality judgment. Our subset selection balances computational feasibility and label distribution fidelity, enabling

efficient neural architecture search within AutoPyTorch while maintaining representative task characteristics.

In addition to GLUE, we evaluate our approach on selected Kaggle datasets that span various text classification domains (e.g., emotion detection, spam filtering), as well as two private corpora in German. These private datasets address real-world classification tasks and introduce additional linguistic and domain-specific variability, allowing us to assess the generalizability of our context-aware AutoML framework across both English and German texts. Detailed dataset statistics are provided in Table 1.

Table 1
Dataset statistics and evaluation metrics. The metric choice reflects task type and class balance.

Dataset	Samples	Labels	Is Balanced	Skew	Metric
GLUE (30% subset)					
QQP	238,572	3	No	2.62	F1-micro
WNLI	255	3	No	2.45	F1-micro
MRPC	1,740	2	No	2.05	F1-micro
CoLA	3,197	3	No	6.34	MCC
RTE	1,730	3	No	2.18	F1-micro
QNLI	34700	3	No	10	Accuracy
SST-2	21012	3	No	10.09	Accuracy
STS-B	2588	3	No	20.88	Pearson
Public and Private Datasets (subset)					
Framing	4,063	2	No	1.67	F1-macro
Troll	517	2	Yes	1.26	Accuracy
Emotion	42,424	2	Yes	1.12	Accuracy
Occupation	10,000	9	No	31.32	F1-micro
Humor	4,000	2	Yes	1.00	Accuracy
Cyber	1665	2	Yes	1.11	Accuracy
BBC	2225	5	Yes	1.32	Accuracy
Math	860	11	No	0.8	F1-Micro
Spam	10455	2	Yes	1.11	Accuracy
Emails	649	2	No	1.5	F1-Micro
Finished Sentence	7973	2	No	4	F1-Micro
Job	2682	2	No	19.63	F1-Micro

4.2. Embedding Method

For all experiments, we used the `all-MiniLM-L6-v2` model from the SentenceTransformers library to generate contextualized text embeddings. The model encodes each input text into a fixed-size dense vector of 384 dimensions.

To ensure a controlled comparison between the baseline and our proposed method, the embedding layer was kept identical across all experimental conditions.

4.3. Model Framework & Search Strategy

We implemented all experiments using the AutoPyTorch framework, leveraging its modular design for deep learning pipelines and extensible search space control. To ensure focus on neural architecture optimization, the traditional machine learning components (e.g., random forests, SVMs) were disabled for our approach. Only deep learning backbones were allowed in the search space.

Specifically, the following backbone architectures were included as candidates:

- **MetaMLP** – a custom MLP architecture whose depth, width, and shape are dynamically adapted based on meta-features such as text length, lexical diversity, number of samples, and class imbalance.
- **Contextual AttentionNet** – a lightweight attention-based model built with multi-head self-attention layers, with structural parameters (e.g., number of heads, embedding dimensions) conditioned on input characteristics.

These architectures were treated as a categorical hyperparameter within the AutoML pipeline, allowing the search process to explore and select the most appropriate model type using Bayesian optimization.

The text-aware version of our pipeline integrates the meta-feature extraction step at the beginning of each AutoPyTorch run. The extracted corpus-level properties are then used to dynamically adapt the hyperparameter search space. Key adaptations include:

- Batch size adjustments based on average sequence length;
- Learning rate and dropout range scaling based on dataset size and class imbalance;
- Architectural shaping (e.g., diamond vs. funnel MLPs) based on input diversity and length variance.

All experiments were constrained to a wall-clock time of 3,000 seconds (approx. 2 hours) and a per-model training time of 600 seconds. We used multi-fidelity optimization via Successive Halving with a training budget ranging from 10 to 100 epochs.

All runs were executed on a single NVIDIA A100 GPU machine with 40 GB of memory, using standard 32-bit floating point precision.

4.4. Baseline Configuration

To establish a fair comparison, we define a strong baseline using the unmodified AutoPyTorch framework and the same text embedding method (MiniLM). In this setting, the hyperparameter search space remains static and is not influenced by any dataset-specific meta-features.

4.5. Results

Tables 2 and 3 summarize the performance of our context-aware AutoML pipeline in comparison to a static AutoPyTorch baseline across 20 text classification datasets. The evaluation was based on four widely used metrics: Accuracy, F1-micro, Matthews Correlation Coefficient

(MCC), and Pearson correlation. These metrics were selected to reflect the characteristics of each task—for example, Accuracy for balanced classification tasks, F1-micro for imbalanced or multi-class problems, MCC for binary grammaticality judgments (e.g., CoLA), and Pearson correlation for sentence similarity (STS-B). Notably, these are also the official evaluation metrics adopted in the GLUE benchmark [33], ensuring compatibility and comparability with prior NLP research.

Overall, our method demonstrates consistent improvements, particularly on tasks characterized by limited training data, class imbalance, or high lexical diversity.

On the GLUE benchmark, our pipeline yields significant gains on several tasks. Notably, WNLI accuracy increases from 17.6% to 54.9%, and SST-2 sees a dramatic rise from 2.1% to 83.4%. These results highlight the effectiveness of our adaptive architecture and regularization mechanisms in low-resource and sentiment-sensitive tasks. We also observe improvements in STS-B, where the Pearson correlation increases from 24.7% to 30.7%. Conversely, slight performance drops are observed in QNLI and CoLA, which may suggest that the current adaptation strategy occasionally introduces suboptimal regularization or architectural choices. For QQP, both the baseline and our pipeline failed to build a viable neural or ensemble model within the computational budget, resulting in fallback to a dummy classifier.

Our pipeline also outperforms the baseline on the majority of Kaggle and private datasets. Substantial gains are observed on *Emails* (from 66.9% to 76.2%), *Troll* (from 52.9% to 56.7%), and *Finished Sentence* (from 79.4% to 81.1%), indicating that context-aware adaptation improves performance in tasks with either noisy data or subtle class distinctions. A slight performance decrease is observed on a few tasks, such as *BBC* and *Occupation*. In the case of *Occupation*, the drop in performance can be attributed to the nature of the dataset: it consists of short, open-ended answers to questions about a person's job, which are then mapped to the top-level labels of the German occupation classification system (KldB). These free-text responses are often terse (e.g., one- or two-word entries like "Technician" or "Sales") and lack sufficient contextual information to support nuanced classification. As a result, the dynamic adaptation mechanisms—designed to adjust architectures and hyperparameters based on richer linguistic cues—have limited room to operate effectively. The scarcity of semantic context may also hinder the effectiveness of embeddings and prevent the model from learning discriminative patterns across fine-grained occupational categories.

Overall, the results support the utility of dynamic search space tailoring across varied domains and textual characteristics.

Table 2
Accuracy (%) comparison between Baseline and Custom Hyperparameter Search across GLUE Datasets.

Dataset	Accuracy/F1-Micro/MCC/Pearson	
	Baseline	**Custom**
CoLA	0.05	-0.2
MRPC	72.7%	74.13%
QNLI	53.3%	50.3%
QQP	49.17%	49.17%
RTE	52.8%	53.5%
SST-2	2.1%	83.4%
STS-B	24.7%	30.72%
WNLI	17.6%	54.9%

Table 3
Prediction performance comparison between Baseline and Custom Hyperparameter Search.

Dataset	Accuracy/F1-Micro (%)	
	Baseline	**Custom**
Occupation	77.9	77.3
BBC	97.3	96.85
Cyber	76.57	77.2
Emails	66.9	76.15
Emotion	87.1	88.3
Framing	69.1	71.3
Humor	91.8	92.87
Math	15.1	16.3
Spam	96.84	96.9
Job	96.46	96.27
Finished Sentence	79.43	81.12
Troll	52.88	56.73

5. Conclusion and Future Works

In this work, we proposed a context-aware AutoML framework that dynamically adapts the hyperparameter search space and neural architecture configurations in response to corpus-level text features. Implemented within the AutoPyTorch ecosystem, our approach integrates dataset-driven meta-feature extraction with a modular design for backbone selection and training parameter control. Experiments across 20 datasets—including subsets of GLUE and diverse public corpora—demonstrate consistent improvements in classification performance, particularly in scenarios with imbalanced classes, small training sets, or high lexical diversity.

By coupling structural and optimization-level decisions to dataset-specific traits, our framework offers a promising direction for more efficient and effective AutoML in NLP. The results validate that even lightweight corpus features (e.g., text length, label imbalance) can yield meaningful adaptations to both model topology and

hyperparameter scheduling.

While our method demonstrates strong empirical gains, several important avenues remain for future research.

First, our current system relies on a limited set of meta-features, such as average text length, vocabulary diversity, and class imbalance. In future work, we aim to extend this analysis to include finer-grained linguistic and structural features such as average sentence length, part-of-speech density, punctuation density, unique character ratio, and readability scores. These features may offer deeper insight into the semantic and syntactic complexity of text, enabling more informed search space adjustments.

Second, while we implement a contextual search space by mapping meta-features to hyperparameter ranges, this process currently uses static, hand-crafted rules. More expressive and structured search spaces could allow hyperparameter relevance and conditionality to adapt dynamically based on dataset characteristics. For instance, certain architecture components or regularization parameters could be activated only when specific linguistic conditions are met, allowing for more flexible and principled adaptation.

Finally, our current fusion strategy for resolving conflicts between feature influences on the same hyperparameter is based on simple heuristics—such as averaging suggested values or intersecting ranges. In future work, we plan to investigate more flexible fusion mechanisms, such as weighting meta-features by importance or learning fusion policies from prior task performance. These improvements could make the contextual adaptation process more scalable, robust, and interpretable across a wide range of text classification tasks.

6. Limitations

Despite the overall effectiveness of our context-aware search space design, several limitations remain.

First, while our system considers multiple meta-features to guide hyperparameter configurations, their influence is combined using static heuristics. This rule-based fusion does not account for potential interactions or conflicts between features, and lacks the flexibility to adapt based on task-specific dynamics or prior performance.

Second, the increased complexity introduced by text-specific search space customization results in higher computational cost. In most cases, we observed longer processing times due to the additional overhead from meta-feature analysis, search space updates, and more expansive architecture evaluations. This may limit the method's applicability in time-constrained or resource-limited environments.

Lastly, our current evaluation focuses solely on classification tasks using moderately sized monolingual datasets. The applicability of our approach to large-scale corpora, multilingual benchmarks, or more complex NLP tasks (e.g., sequence labeling or generation) remains unexplored.

References

[1] W. Lam, K.-Y. Lai, A meta-learning approach for text categorization, in: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, 2001, pp. 303–309.

[2] J. G. Madrid, H. J. Escalante, E. Morales, Meta-learning of textual representations, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2019, pp. 57–67.

[3] H. Jin, F. Chollet, Q. Song, X. Hu, Autokeras: An automl library for deep learning, Journal of machine Learning research 24 (2023) 1–6.

[4] X. Shi, J. Mueller, N. Erickson, M. Li, A. J. Smola, Benchmarking multimodal automl for tabular data with text fields, arXiv preprint arXiv:2111.02705 (2021).

[5] M.-A. Zöller, M. F. Huber, Benchmark and survey of automated machine learning frameworks, Journal of artificial intelligence research 70 (2021) 409–472.

[6] S. Saleem, S. Kumarapathirage, A systematic review of automl for text classification: From theory to practice (2023).

[7] P. Safikhani, D. Broneske, Enhancing autonlp with fine-tuned bert models: an evaluation of text representation methods for autopytorch, Available at SSRN 4585459 (2023).

[8] P. Safikhani, D. Broneske, Automl meets hugging face: Domain-aware pretrained model selection for text classification, in: Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 4: Student Research Workshop), 2025, pp. 466–473.

[9] J. C. Gomez, S. Hoskens, M.-F. Moens, Evolutionary learning of meta-rules for text classification, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2017, pp. 131–132.

[10] M. J. Ferreira, P. Brazdil, Workflow recommendation for text classification with active testing method, in: Workshop AutoML, 2018.

[11] R. Desai, A. Shah, S. Kothari, A. Surve, N. Shekokar, Textbrew: Automated model selection and hyperparameter optimization for text classification, In-

ternational Journal of Advanced Computer Science and Applications 13 (2022).

[12] C. Thornton, F. Hutter, H. H. Hoos, K. Leyton-Brown, Auto-weka: Combined selection and hyperparameter optimization of classification algorithms, in: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, 2013, pp. 847–855.

[13] M. Feurer, F. Hutter, Automated machine learning, Cham: Springer (2019) 113–134.

[14] M. Wistuba, N. Schilling, L. Schmidt-Thieme, Hyperparameter search space pruning–a new component for sequential model-based hyperparameter optimization, in: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part II 15, Springer, 2015, pp. 104–119.

[15] E. Öztürk, F. Ferreira, H. Jomaa, L. Schmidt-Thieme, J. Grabocka, F. Hutter, Zero-shot automl with pretrained models, in: International Conference on Machine Learning, PMLR, 2022, pp. 17138–17155.

[16] Y. Wang, Y. Yang, Y. Chen, J. Bai, C. Zhang, G. Su, X. Kou, Y. Tong, M. Yang, L. Zhou, Textnas: A neural architecture search space tailored for text representation, in: Proceedings of the AAAI conference on artificial intelligence, volume 34, 2020, pp. 9242–9249.

[17] J. Xu, X. Tan, R. Luo, K. Song, J. Li, T. Qin, T.-Y. Liu, Nas-bert: Task-agnostic and adaptive-size bert compression with neural architecture search, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 1933–1943.

[18] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers), 2019, pp. 4171–4186.

[19] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, Q. V. Le, Xlnet: Generalized autoregressive pretraining for language understanding, Advances in neural information processing systems 32 (2019).

[20] L. Zimmer, M. Lindauer, F. Hutter, Auto-pytorch: Multi-fidelity metalearning for efficient and robust autodl, IEEE transactions on pattern analysis and machine intelligence 43 (2021) 3079–3090.

[21] Y. Li, Y. Shen, W. Zhang, Y. Chen, H. Jiang, M. Liu, J. Jiang, J. Gao, W. Wu, Z. Yang, et al., Openbox: A generalized black-box optimization service, in: Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining, 2021, pp. 3209–3219.

[22] A. McCartney, S. Hensman, L. Longo, How short is a piece of string?: the impact of text length and text augmentation on short-text classification accuracy (2017).

[23] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, E. Hovy, Hierarchical attention networks for document classification, in: Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies, 2016, pp. 1480–1489.

[24] I. Beltagy, M. E. Peters, A. Cohan, Longformer: The long-document transformer, arXiv preprint arXiv:2004.05150 (2020).

[25] M. Monteiro, C. K. James, M. Kloft, S. Fellenz, Characterizing text datasets with psycholinguistic features, in: Findings of the Association for Computational Linguistics: EMNLP 2024, 2024, pp. 14977–14990.

[26] M. Sokolova, Big text advantages and challenges: classification perspective, International Journal of Data Science and Analytics 5 (2018) 1–10.

[27] C. Zhang, S. Bengio, M. Hardt, B. Recht, O. Vinyals, Understanding deep learning requires rethinking generalization, arXiv preprint arXiv:1611.03530 (2016).

[28] V. Vapnik, Statistical Learning Theory now plays a more active role: after the general analysis of learning processes, the research in the area of synthesis of optimal algorithms was started. These studies, however, do not belong to history yet. They are a subject of today's research activities., Ph.D. thesis, These studies, however, do not belong to history yet. They are a subject of ..., 1998.

[29] T. Domhan, J. T. Springenberg, F. Hutter, Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves., in: IJCAI, volume 15, 2015, pp. 3460–8.

[30] P. Probst, M. N. Wright, A.-L. Boulesteix, Hyperparameters and tuning strategies for random forest, Wiley Interdisciplinary Reviews: data mining and knowledge discovery 9 (2019) e1301.

[31] J. L. Leevy, T. M. Khoshgoftaar, R. A. Bauder, N. Seliya, A survey on addressing high-class imbalance in big data, Journal of Big Data 5 (2018) 1–30.

[32] S. Uddin, H. Lu, Dataset meta-level and statistical features affect machine learning performance, Scientific Reports 14 (2024) 1670.

[33] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, S. R. Bowman, Glue: A multi-task benchmark and analysis platform for natural language understanding, arXiv preprint arXiv:1804.07461 (2018).