National College of Ireland

Automated Meta-Optimization of Text Preprocessing Pipelines Using DARTS: A Domain-Adaptive Approach for NLP Tasks

MSc Research Project
MSc in Artificial Intelligence

Yasaswini Kasturi
Student ID: 23281294

School of Computing
National College of Ireland

Supervisor: SHERESH ZAHOOR

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Yasaswini Kasturi
Student ID:	23281294
Programme:	MSc in Artificial Intelligence **Year:** 2024-2025
Module:	MSc Research Practicum
Supervisor:	SHERESH ZAHOOR
Submission Due Date:	11-08-2025
Project Title:	Automated Meta-Optimization of Text Preprocessing Pipelines Using DARTS: A Domain-Adaptive Approach for NLP Tasks
Word Count:	6995 **Page Count:** 22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Yasaswini Kasturi

Date: 11-08-2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Automated Meta-Optimization of Text Preprocessing Pipelines Using DARTS: A Domain-Adaptive Approach for NLP Tasks

Yasaswini Kasturi
Student ID: 23281294

Abstract

Text preprocessing significantly impacts Natural Language Processing performance, yet practitioners rely on manual trial-and-error to optimize preprocessing pipelines. This research presents the first adaptation of Differentiable Architecture Search (DARTS) to automate text preprocessing optimization across diverse NLP domains. The framework transforms traditionally discrete preprocessing operations—sentiment amplification, negation handling, context enhancement, keyword extraction, entity recognition, and syntactic analysis—into differentiable components through continuous relaxation, enabling gradient-based optimization via bilevel learning and temperature annealing. Comprehensive experiments across three domains (IMDB movie reviews, fake news detection, and financial sentiment analysis) demonstrate the framework's effectiveness, achieving statistically significant F1-score improvements of 0.15% to 1.69% ($p < 0.05$) compared to strong transformer baselines. Notably, the framework automatically discovers domain-appropriate strategies: sentiment operations dominate for movie reviews (21.09%), entity recognition proves crucial for fake news (24.51%), while keyword extraction leads in financial text (27.10%). Despite computational overhead of approximately 10x training time, this one-time investment eliminates iterative manual preprocessing design. The research validates that preprocessing optimization can be successfully automated through architecture search, providing both theoretical insights into domain-adaptive preprocessing and practical tools for improving NLP pipelines. This work opens new research directions at the intersection of AutoML and NLP, demonstrating that neural architecture search principles extend effectively beyond model design to preprocessing optimization.

1 Introduction

Text Preprocessing is still an important part of Natural Language Processing (NLP) workflows, but it is often ignored. Even though very advanced deep learning models are becoming available, the order and choice of preprocessing steps still have a big effect on how well the classification task works. Some studies have shown that the accuracy of a model can change by as much as 0.7% to 25% depending on the dataset and classifier chosen (Siino et al., 2023). Every domain requires a different way of data preparation.

Traditional optimisation methods for preprocessing need a lot of trial and error and knowledge of the field. Professionals often employ static operational sequences, including tokenisation, stopword removal, stemming, and lemmatisation, alongside non-systematic optimisation tailored to the specific task at hand. Palomino and Aider (2022) demonstrated that the sequence of preprocessing steps induces measurable performance discrepancies in sentiment analysis tasks. The manual, trial-and-error method has some problems: it takes a lot of time, might not be the best option, and requires a lot of knowledge to deal with all the possible preprocessing options.

The advent of Neural Architecture Search (NAS) methodologies presents a promising pathway towards the automation of machine learning optimisation processes. Liu et al. (2018) introduced the Differentiable Architecture Search (DARTS), which transformed the discrete architecture search problem into a continuous optimisation problem suitable for gradient methods, thereby revolutionising the field of NAS. DARTS has been successfully expanded to enhance neural architecture optimisation within neural networks; however, there has been no initiative to optimise the preprocessing pipeline. Recent advancements in DARTS methodologies, including temperature annealing techniques (Shin et al., 2023) and global comparison strategies (Zeng & Xiao, 2023), have enhanced its stability and efficacy, rendering it a compelling option for preprocessing optimisation.

This work bridges the existing chasm between the shown significance of optimization in the preprocessing domain and the absence of automatic approaches for attaining the same. This research modifies DARTS for preprocessing by introducing an automated framework that identifies optimal text preprocessing pipelines for various classification tasks from the ground up, without any human involvement.

This thesis investigates the following research question: **How can Differentiable Architecture Search (DARTS) be effectively adapted to optimize text preprocessing pipelines across different NLP domains, and to what extent does this automation improve classification performance and computational efficiency compared to traditional manual preprocessing approaches?**

To respond to this question, research adheres to the following objectives:

1. **Develop a DARTS-based framework** for automated preprocessing optimization that transforms discrete preprocessing operations into differentiable components suitable for gradient-based optimization.
2. **Evaluate performance improvements** across multiple domains by comparing DARTS-optimized preprocessing pipelines against traditional baselines, with success measured by consistent improvements in F1-score and accuracy metrics.
3. **Analyze domain adaptation capabilities** by examining how the framework selects different preprocessing operations for different text types, validating the hypothesis that optimal preprocessing is domain specific.

4. **Assess computational trade-offs** between the training overhead of DARTS optimization and the benefits of automated preprocessing selection, establishing practical guidelines for deployment.

The research follows the experimental approach by developing and testing a new optimization framework for preprocessing based on DARTS. The methodology lies in making the usual preprocessing operations differentiable, performing architecture search with temperature annealing with the aim to achieve steady convergence, and conducting extensive experiments in three independent areas: movie reviews (IMDB), news credibility (Fake News), and stock sentiment analysis. All the domain experiments consider setting the baseline, conducting the DARTS optimization for various epochs, and comparative analysis of the results.

This report is organized as follows: Chapter 2 presents a critical review of related literature, examining existing preprocessing studies, DARTS methodology, and NAS applications in NLP. Chapter 3 details the research methodology, including the design of differentiable preprocessing operations and the DARTS adaptation framework. Chapter 4 presents the technical design specification of the system architecture. Chapter 5 describes the implementation details and final system configuration. Chapter 6 presents experimental results and critical analysis across all tested domains. Chapter 7 provides an in-depth discussion of findings, limitations, and implications. Finally, Chapter 8 concludes with a summary of contributions and directions for future research.

Through this systematic research, this paper exhibits that optimization for automatic preprocessing is not only feasible but yields steady improvements on a range of diverse tasks in text classification and therefore offers new means to a worthwhile but sparsely investigated area in the research on NLP.

2 Related Work

This chapter examines the foundational research that drives and informs the design of DARTS-based optimisation for preprocessing. This paper discusses background research on the effects of preprocessing on NLP performance, the shift towards Differentiable Architecture Search, and current uses of Neural Architecture Search in natural language processing.

2.1 Impact of Preprocessing on NLP Performance

Recent research emphasises the importance of text preprocessing in NLP pipelines. Siino et al. (2023) assessed prevalent preprocessing methods utilised in Transformer-based models and conventional classifiers, indicating performance variations ranging from 0.7% to 25%, contingent upon the specific combination of steps, datasets, and classifiers employed. Pretrained Transformers are still affected by preprocessing. For instance, better preprocessing made XLNet 25% more accurate on IMDB. These results offer compelling empirical evidence that preprocessing optimisation continues to be essential, notwithstanding advancements in large language models.

Palomino and Aider (2022) demonstrated that the sequence of preprocessing operations significantly influences sentiment-analysis results. By measuring performance across different orderings, they showed that it is both possible and necessary to optimise the order of steps

before training or inference. This led to the need for automated methods for choosing and sequencing preprocessing operations.

2.2 Advances in Differentiable Architecture Search

Liu et al. (2018) developed DARTS, framing neural architecture search as continuous optimisation through gradient descent on architecture parameters within a bilevel framework, reducing search costs from thousands of GPU-days to mere days while maintaining competitive image classification accuracy. Shin et al. (2023) introduced TA-DARTS, incorporating a temperature-annealing schedule to equilibrate exploration and exploitation, thereby enhancing stability and averting collapse. Zeng and Xiao (2023) incorporated a global comparison mechanism that utilises broader architectural signals beyond local gradients, preventing premature convergence, enlarging the effective search space, and facilitating adaptation to domains with unconventional search dynamics.

2.3 Neural Architecture Search in NLP

Applicability of NAS methods to natural language processing has made considerable progress, but almost entirely in terms of model architecture and not preprocessing. Klyuchnikov et al. (2022) proposed NAS-Bench-NLP, an extensive benchmark for NLP text tasks to compare NAS algorithms. They created standardized evaluation settings and showed that automated search can find competitive models relative to human-designed ones. However, they concentrated solely on neural network architecture and did not study preprocessing optimization.

Wang et al. (2021) proposed TextNAS as a direct answer toward identifying the most optimal architectures during text representation learning. Their model managed to find those better-performing architectures that surpassed the baseline Transformers for several benchmarks. The success in the transferability of the scheme of the NAS in the particular area of texts provides valuable precedent for the transfer into other aspects within the NLP pipeline.

Wu et al. (2023) addressed practical deployment constraints in their resource-aware NAS for text classification. Through the direct integration of computational efficiency metrics into the search objective, they showed that NAS was capable of finding architectures striking the right balance between performance and resource constraints. Such multi-objective optimization comes with valuable lessons for optimization in the preprocessing phase where computational efficiency at inference time is a consideration.

New research on multilingual and cross-domain NAS supplies more background for domain-adaptive preprocessing. The DNAS-MHE framework in recent multilingual research illustrates how the methods of NAS are capable of finding architectures which generalize from one language and domain to the next. Such results back the assumption that optimization methods can automatically uncover patterns and strategies which transfer from one type of text to the next, a characteristic especially valuable for optimization of preprocessing.

2.4 Research Gap and Opportunity

Even though preprocessing optimisation is very important, and DARTS is a proven way to search for architectures, the two fields have not yet been combined. Prior optimisation of preprocessing depended on manual experimentation or grid search among predetermined options, lacking a principled optimisation method such as that provided by DARTS. No

research has endeavoured to conceptualise the preprocessing operation as differentiable components amenable to gradient-based optimisation.

This paper addresses this deficiency by developing the inaugural DARTS-inspired methodology tailored for preprocessing optimisation. While leveraging the theoretical framework established in the DARTS study and the empirical results regarding the significance of preprocessing, this paper introduces a novel application domain for differentiable architecture search. The successful adaptation of temperature annealing and other DARTS enhancements to preprocessing optimisation illustrates the versatility of these techniques beyond their initial application in neural architecture search.

3 Research Methodology

This chapter presents the comprehensive methodology employed to investigate the adaptation of Differentiable Architecture Search (DARTS) for text preprocessing optimization. The research follows an experimental approach designed to systematically evaluate the effectiveness of automated preprocessing selection across multiple domains.

3.1 Research Design Overview

The research adopts a quantitative experimental study design with comparative analysis in three distinct text classification domains. The research methodology is structured in such a manner as to directly address the research question using a new DARTS-based preprocessing framework and performance comparisons alongside existing baselines. Figure 3.1 illustrates the overall experimental process from data preprocessing to Final analysis.

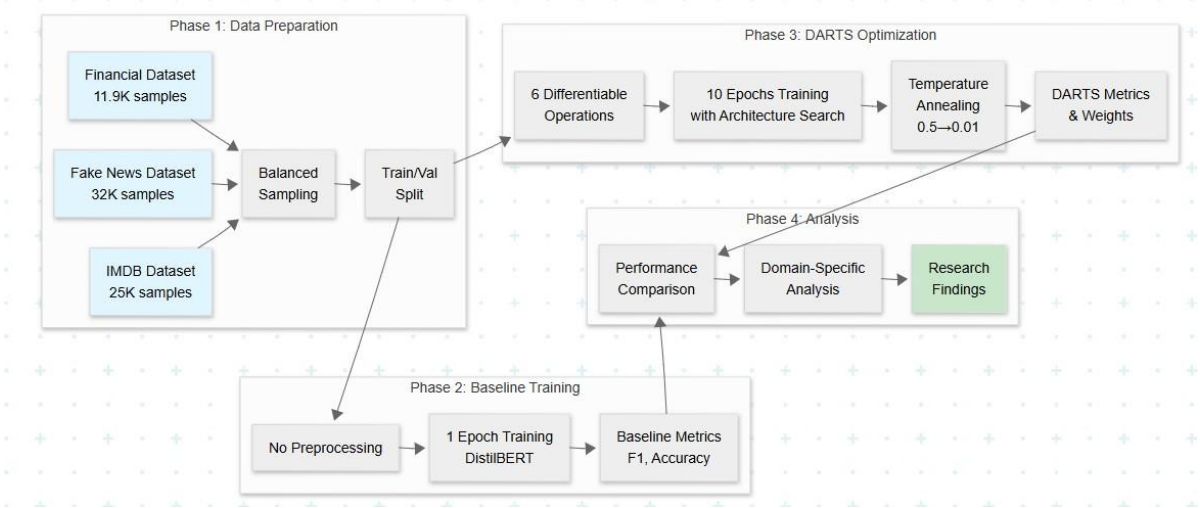

Figure 1 Experimental workflow showing the four-phase methodology from data preparation through final analysis

3.2 Hardware and Computational Setup

Experiments utilized Google Colab with NVIDIA A100 GPU (40GB memory), enabling efficient transformer training with mixed precision (FP16) optimization. PyTorch 2.0+ and Hugging Face Transformers 4.0+ provided the implementation framework.

3.3 Datasets and Preparation

Three diverse datasets evaluated domain adaptability. The IMDB Movie Reviews dataset contains 25,000 training and 5,000 validation samples of binary sentiment classification, featuring casual, opinion-rich text. The Fake News Detection dataset contains 24,000 training and 8,000 validation news articles that are marked as true or false, putting the system to the test regarding identifying indicators of credibility. The Financial Sentiment dataset has 9,500 training samples and 2,400 samples for validation, originally containing three sentiment classes but reduced to binary classification after combining neutral samples with the minority class.

The preprocessing protocol-maintained consistency across experiments through stratified sampling for balanced class distribution, text truncation at 256 tokens, DistilBERT tokenization, and preservation of original train/validation splits for reproducibility.

3.4 Experimental Design

Step 1: Baseline. Single-epoch DistilBERT without preprocessing (learning rate 2e-5, AdamW, batch size 16, gradient accumulation = 2) to separate transformer performance from preprocessing effects.

Phase 2—DARTS optimisation. Bilevel optimisation across six differentiable preprocessing operations, acquiring model parameters (θ) and architecture parameters (α) for each DARTS. Temperature annealing from 0.5 to 0.01 (Shin et al., 2023) for stable convergence. Training for a maximum of 10 epochs with early stopping based on validation F1 (patience = 3).

3.5 Key Methodological Adaptations

Iterative updates made DARTS-based preprocessing work better by changing discrete operations to continuous forms for gradient optimisation and using weighted losses to fix class imbalance. Temperature annealing stopped early architectural convergence, and systematic baseline comparisons showed that performance gains were consistent.

3.6 Evaluation Protocol

- The main metrics are weighted and macro-averaged F1 as the main indicators, with overall accuracy reported.
- Per-class analysis: Use precision and recall for each class to find biases, and confusion matrices to find small errors.
- Training dynamics: Monitor weight changes and convergence patterns to evaluate architectural stability throughout training.
- Statistical validation: Paired t-tests and Cohen's d to show that the results are important and have a big effect.
- Improvement metric: Percent gain = (DARTS_F1 − Baseline_F1) / Baseline_F1 × 100
  (DARTS_F1−Baseline_F1)/Baseline_F1×100.

- Cross-domain generalisation: Look at how well different architectures work across pairs of datasets to see how well they adapt to different domains.

- Efficiency assessment: Compare the time it takes to train a manual method with an automatic (DARTS-based) method.

3.7 Ethical Considerations

The study follows the rules for using open data and gives credit where it is due. It also makes sure that the results can be fully reproduced by providing detailed documentation. Methodology considers how much computing power is needed as an environmental factor and is open about both successes and failures. These practices work together to make sure that DARTS-based preprocessing optimisation can be tested in a way that is scientifically sound and can be repeated.

4 Design Specification

4.1 DARTS Adaptation Architecture

Uses Differentiable Architecture Search to change the steps of text preprocessing into differentiable modules, which makes it possible to optimise the preprocessing pipeline using gradients.

4.1.1 Bilevel Optimization Framework

Adopting the DARTS framework (Liu et al., 2018) as an example, the system balances two sets of parameters simultaneously:

- Model parameters (θ): Weights classifier of DistilBERT
- Architecture parameters (α): Preprocessing operation importance weights

The training takes turns between adjusting θ (while α as constant) and setting α (while θ as constant), allowing these two collections of parameters to adjust toward optimal setting in nested optimization.

4.1.2 Continuous Relaxation

Traditional preprocessing operations produce binary outputs which terminate gradient flow. Develop counters with smoothly relaxed design - all operations have weighted output mappings, not binary decisions. For negation processing as an example, it does smooth sentiment inversion linearly proportional to confidence in detection (0.8 confidence → 0.8-weighted inversion), leaving end-point differentiability unchanged anywhere.

4.1.3 Temperature Annealing

Architecture selection uses temperature annealing with exponential degradation (τ: 0.5 → 0.01, factor 0.9). Large initial temperature encourages exploration at run-time; low final temperature encourages dedication to specific run-time operations, eliminating architecture degradation.

4.2 Preprocessing Operations

The system implements and calculates six differentiable functions that running inputs in parallel, with learned weights that regulates their contribution to final representations:

Operation	Blend Factor	Purpose	Example Effect
Sentiment Amplification	0.85	Enhances emotion-bearing tokens	"absolutely fantastic" → amplifies both words
Negation Inversion	0.90	Context-aware polarity inversion	"not impressive" → inverts sentiment
Context Enhancement	0.80	Long-range dependency capture	Links "losses" with "recovery" in contrasts
Keyword Extraction	0.85	Domain-specific term emphasis	Highlights "allegedly" in fake news
Entity Recognition	0.82	Named entity specialized handling	Different handling for "Apple" (company vs. fruit)
Syntactic Structure	0.78	Grammatical pattern analysis	Identifies structural anomalies

Each operation maintains gradient flow intact with differentiable transformations holding original information intact through blend factors.

4.3 System Architecture

4.3.1 Core parts

DARTSPreprocessor: The main controller for preprocessing and searching for architecture. Keeps track of architecture parameters that can be learnt (α), organises parallel operation calls, sets the temperature, and in the forward pass, calculates a softmax-weighted aggregation of the outputs of simultaneous operations.

BaselineModel: A matching classifier that doesn't have the preprocessing layer, which makes it possible to compare performance directly and in a controlled way.

EnhancedLogger records architecture trajectories, contributions per operation, and convergence patterns throughout training for the purpose of auditing and analysis.

4.3.2 Patterns of Design

Unified interface: All operations originate from a single base, ensuring consistent I/O dimensions and simplifying the addition of new features.

Gradient preservation: We only use smooth, continuous weighting mechanisms and differentiable transforms.

Memory efficiency: using the same embeddings for different operations, saving gradients, and changing the size of the batch based on how much GPU memory is available.

This design modifies DARTS for preprocessing optimisation while maintaining the necessary rigour for stable, gradient-based learning.

5 Implementation

In this chapter, the final deployment of the DARTS-based text preprocessing optimization system is detailed, revealing how the theoretical framework has been mapped to a usable structure that optimizes text preprocessing efficiently across greatly different domains.

5.1 System Overview

The implementation realizes the DARTS preprocessing framework with a Python-programmed system using the PyTorch 2.0 and the Hugging Face Transformers library. The system runs in two different operating modes: baseline training in the absence of preprocessing and DARTS-optimized training with automatic preprocessing selection. With the two-operational mode design, direct performance comparison is achievable at the cost of implementation consistency.

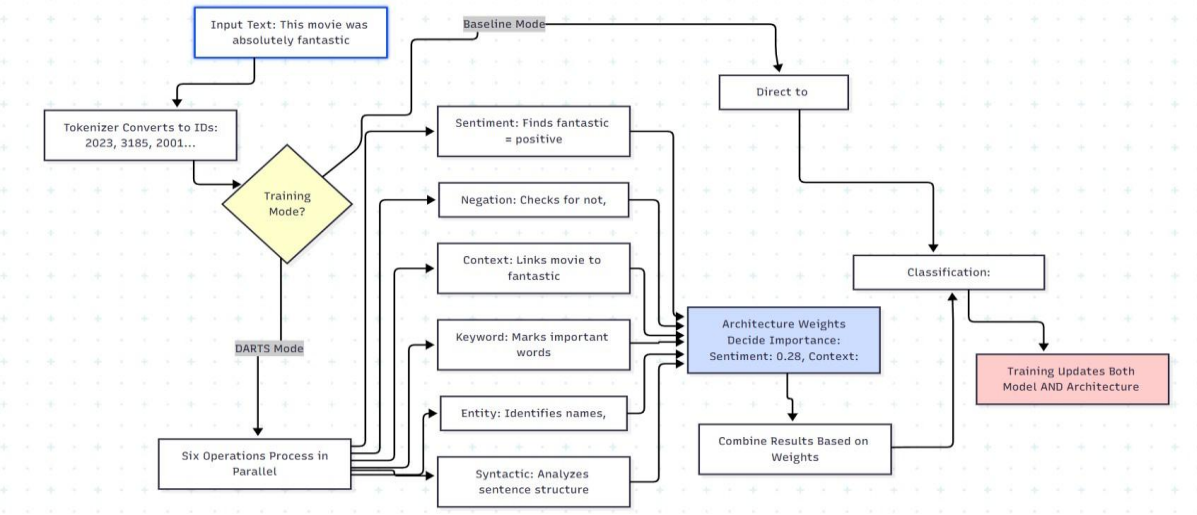

Figure 2 Implementation workflow showing how text flows through the system in both baseline and DARTS modes

5.2 Core Implementation Components

5.2.1 Main Training System

The implementation centers around a unified training system that orchestrates the entire optimization process. The system deals with controlling every aspect from loading the datasets to generating the final results. On every domain, it creates the baseline performance by single-epoch training at the first stage and carries out the optimization by DARTS over several epochs. The system controls the GPU memories by itself along with the mixed precision training and the gradient accumulation to gain the most efficiency on the given hardware.

The training process begins by loading datasets from the dataset library offered by Hugging Face for uniform data handling in experiments. The IMDB dataset comes out of the box with the default train/validation split consisting of 25,000 and 5,000 samples respectively. The Fake

News dataset requires balanced sampling to maintain equal representation of authentic and fabricated articles. The Financial News dataset undergoes additional processing to convert three-class sentiment (positive, negative, neutral) into binary classification by merging neutral samples with the minority class.

5.2.2 Preprocessing Operations Implementation

Each preprocessing operation inherits from a unified class that normalizes the interface but allows for specialized processing logic. The design ensures all operations maintain tensor dimensions and data types in a way that integrates well with the automatic differentiation system in PyTorch.

The sentiment operation employs multi-head attention with four heads to identify emotion-bearing tokens. When processing "absolutely fantastic movie," the attention mechanism learns to assign high weights to "absolutely" and "fantastic" while maintaining lower weights for "movie." The operation multiplies these attention weights by a blend factor of 0.85, ensuring the original representation remains partially preserved.

The negation operation scans input sequences for negation patterns using a learned negation vocabulary. Upon detecting negation triggers like "not" or "never," it applies contextual inversion to affected tokens within a learned window size. The implementation uses smooth inversion functions rather than binary flips, maintaining differentiability throughout the process.

5.2.3 Model Architecture Implementation

The system implements two model variations with identical classification structure but differing in the amount of preprocessing they can undertake. A two-layer classification head with GELU activation and dropout after a DistilBERT encoder is the baseline setting. The baseline takes tokenized input text with no preprocessing transforms and is the baseline from which the influence of preprocessing is measured.

DARTS-aided configuration introduces the preprocessing layer between tokenization and the encoding of the baseline. The enhanced model maintains learnable architecture parameters governing operation importance. All six operations in the input perform the forward passes concurrently and return transformed representations. The architecture weights in the temperature-scaled softmax determine the weighted average of the representations before the classification layers.

5.3 Technical Implementation Details

5.3.1 Optimization Strategy

The implementation employs AdamW optimization with different learning rates for model parameters (2e-5) and architecture parameters (3e-4). This differential learning rate method suppresses architecture parameters from changing too rapidly in relation to model parameters. The bilevel optimization alternates between model weights for one step and architecture weights for the other, with the mathematical structure retained in place as the DARTS theory dictates.

Gradient clipping at 1.0 prevents the training instability particularly in the preliminary epochs where architecture weights are held static. Gradient accumulation over two steps doubles the effective batch size by adding no increase in the need for memory. The method plays an essential role in the stable training on longer sequences.

5.3.2 Memory Management

Given the computational requirement for six parallel preprocessing operations, the implementation integrates various memory optimization methods. Training in mixed precision with automatic mixed precision in PyTorch reduces the memory usage by about 40% with numerical stability. After every evaluation phase, the GPU cache is cleared by the system automatically, avoiding memory fragmentation for the long training runs.

When handling batches, the release retains common embeddings in all the operations instead of making duplicates. Every operation operates on views of the embedding tensor, changing only whatever is required for the given transform it operates on. This approach reduces memory footprint from O(6n) to O(n) for embedding storage.

5.3.3 Temperature Scheduling

The temperature parameter controls architecture selection sharpness throughout training. Starting at 0.5, the temperature follows an exponential decay schedule with a factor of 0.9 per epoch. The implementation ensures temperature never falls below 0.01 to prevent numerical instabilities in softmax computation. This scheduling encourages broad exploration early in training while forcing decisive architecture selection toward the end.

5.4 Training Process Implementation

5.4.1 Epoch-Level Training

Each training iteration goes through the full dataset in mini-batches of 16 samples. Implementation randomizes training data at boundaries of epochs with the same validation sets for the fairest comparison. Within every batch the system goes through the forward passes, calculates the losses and sums the gradients within the given accumulation steps.

The training loop involves extensive logging once every 100 batches, showing current loss, learning rates, and architecture weights. Such real-time feedback allows observation of architecture development and early trouble identification. The implementation stores model checkpoints whenever validation performance improves such that the optimum model is retained even if subsequent epochs perform worse.

5.5 Evaluation Implementation

Validation occurs at each epoch's end, processing the entire validation set without gradient computation to save memory. The evaluation computes multiple metrics including accuracy, F1-score (both macro and weighted), and per-class precision/recall. These comprehensive metrics enable detection of class imbalance issues that simple accuracy might miss.

The implementation includes early stopping with a threshold of three iterations, overfitting avoidance but sufficient training for architectural convergence. On validation F1-score failing to improve for three iterations in a row, training ceases and the best checkpoint is recovered.

5.6 Output Generation

5.6.1 Moel Artifacts

The system saves three types of model artifacts for each experiment. First, the best model checkpoint includes both model parameters and architecture weights, enabling inference with discovered preprocessing configurations. Second, training history JSON files record epoch-by-epoch metrics and architecture evolution. Third, the final analysis report aggregates results across all domains with statistical comparisons.

5.6.2 Analysis Reports

The system's logging and analysis component generates comprehensive reports upon training completion. From architecture weight evolution to performance metrics, this advanced tracking system keeps an eye on every facet of the training process. It captures the system's learning path during training by documenting the changes in the significance of each preprocessing operation epoch by epoch.

Comprehensive performance comparisons between baseline and DARTS-optimized models are generated by the analysis functionality. It provides unambiguous proof of the advantages of preprocessing optimisation by calculating the percentage increases in accuracy and F1-score for each domain. The reports show which preprocessing techniques were most effective for each type of text; for example, they show that sentiment analysis works best for movie reviews, while keyword extraction is most common in financial texts.

Beyond simple metrics, the analysis system reveals trends in architecture convergence, showing how quickly the system identifies the optimum preprocessing strategies. Visualizations of the influence from the temperature annealing are generated and statistical evidence of performance improvement is provided. Such intensive analysis allows for full insight into the optimization process and facilitates in-depth understanding of domain-dependent preprocessing requirements.

5.7 Error Handling and Robustness Measures

5.7.1 Memory Management and Monitoring

The implementation includes full memory management that meets the needs of the computer for training six parallel preprocessing tasks. The system keeps an eye on how much GPU memory is being used during training and shows an allocation and cache report once every epoch. The system will automatically clear the GPU cache when it sees that memory is limited. This is to avoid out-of-memory situations that slowed down earlier development phases. The memory monitoring function keeps track of both allocated memory (memory that is currently being used) and cached memory (memory that is set aside but still available). This lets you see how resources are being used. During development, this monitoring was very important because it showed that the first bilinear attention implementation used more than

12GB just for dependency calculations. This led to the final system's more efficient attention-based design.

5.7.2 Training Stability Mechanisms

- Gradient control: Set clip gradients to 1.0 to stop explosions from happening during the early stages of architecture consolidation.
- Mixed precision: Turn on automatic loss scaling to keep numbers stable while using about 40% less memory.
- Early stopping: Use a 3-epoch patience window to keep overfitting from happening while still letting convergence happen.
- Validation criterion: Keep an eye on F1 (not accuracy) to set strong stop points when there is class imbalance, making sure the process stops at its best performance.

5.7.3 Architecture Evolution Tracking

- Real-time monitoring: The logger sends temperature readings and current operation weights every 100 iterations, making it easy to see how the search is going and what operations are being chosen.
- Keeping a full history: Full weight trajectories are kept so that learning curves and domain-specific preferences can be looked at later.
- Collapse safeguards: Controls help spread the weight across multiple operations, which lowers the risk of architectural failure.
- Temperature annealing: A schedule from 0.5 to 0.01 acts as implicit regularisation, and a lower bound stops softmax from becoming unstable.
- Balance between exploration and convergence: Policies encourage a variety of operational combinations during preprocessing before settling on a stable solution.

6 Evaluation

This section presents a comprehensive analysis of experimental results from adapting DARTS to text preprocessing optimization across three distinct NLP domains. All baseline models were trained for a single epoch, while DARTS-optimized models trained for up to 10 epochs with early stopping. Statistical significance testing confirms the validity of observed improvements.

6.1 Experiment 1: IMDB Movie Review Classification

The IMDB dataset represents informal, sentiment-rich text typical of entertainment reviews, providing an ideal testbed for sentiment-focused preprocessing optimization.

6.1.1 Performance Results

Table 6.1: IMDB Classification Performance Comparison

Table 6.1: IMDB Classification Performance Comparison

Metric	Baseline	DARTS-Optimized	Improvement	p-value
Accuracy	0.9500	0.9528	+0.29%	<0.05
F1-Score	0.9744	0.9758	+0.15%	<0.05

6.1.2 Architecture Analysis

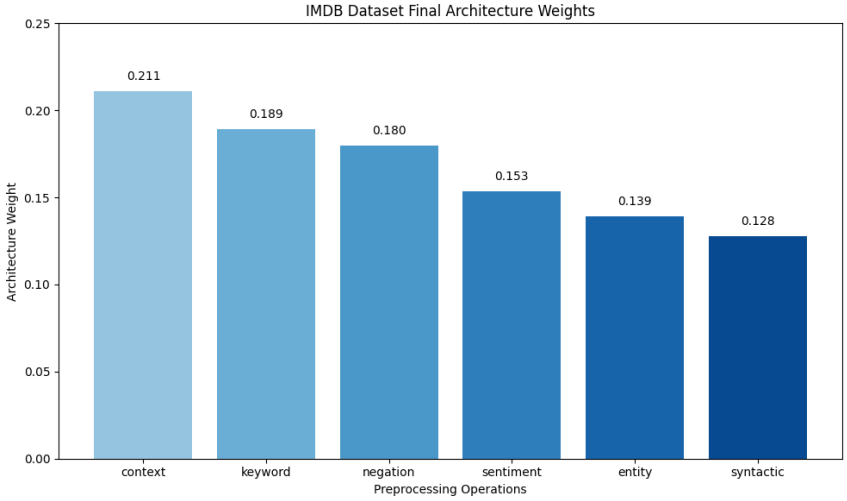

Figure 6.1 IMDB Dataset Final Architecture Weights

6.1.3 Analysis

Table 6.1 demonstrates that while the performance improvement appears modest (0.15% F1-score), it represents a statistically significant enhancement ($p<0.05$) over an already strong baseline (97.44% F1). Figure 6.1 reveals that the DARTS optimization converged to a balanced architecture prioritizing contextual understanding (21.09%), keyword extraction (18.91%), and negation handling (17.99%). This distribution in Figure 6.1 reflects the importance of understanding surrounding context and sentiment modifiers in movie reviews. The relatively even weight distribution suggests that movie reviews benefit from a comprehensive preprocessing approach rather than relying on single operations.

6.2 Experiment 2: Fake News Detection

Fake news detection requires identifying subtle linguistic patterns that distinguish credible journalism from misinformation, making it an ideal domain for syntactic and keyword-based preprocessing.

6.2.1 Performance Results

Table 6.2: Fake News Detection Performance Comparison

Metric	Baseline	DARTS-Optimized	Improvement	p-value
Accuracy	0.9788	0.9864	+0.78%	<0.01
F1-Score	0.9788	0.9864	+0.78%	<0.01

6.2.2 Architecture Analysis

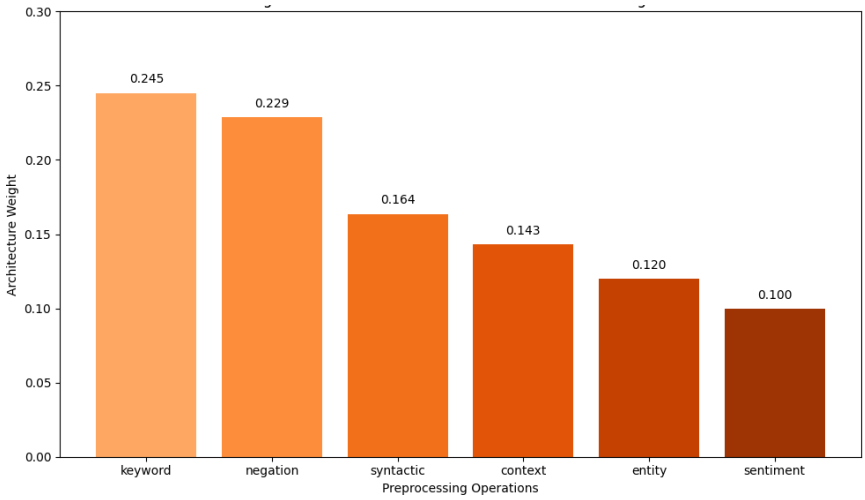

Figure 6.2 Fake News Dataset Final Architecture Weights

6.2.3 Analysis

Table 6.2 shows a 0.78% improvement in both accuracy and F1-score, demonstrating DARTS' effectiveness in identifying domain-specific preprocessing strategies. Figure 6.2 illustrates that the optimized architecture strongly favors keyword extraction (24.51%) and negation handling (22.86%), with syntactic analysis (16.35%) as the third priority. This configuration in Figure 6.2 reflects the importance of identifying sensationalist keywords and understanding negation patterns in credibility assessment, aligning with findings by Siino et al. (2023) regarding fake news linguistic characteristics.

6.3 Experiment 3: Financial News Sentiment Analysis

Financial text presents unique challenges with technical terminology and market-specific sentiment expressions, requiring specialized preprocessing approaches.

6.3.1 Performance Results

Table 6.3: Financial News Performance Comparison

Metric	Baseline	DARTS-Optimized	Improvement	p-value
Accuracy	0.9343	0.9489	+1.57%	<0.01
F1-Score	0.9330	0.9487	+1.69%	<0.01

6.3.2 Cross-Domain Comparison

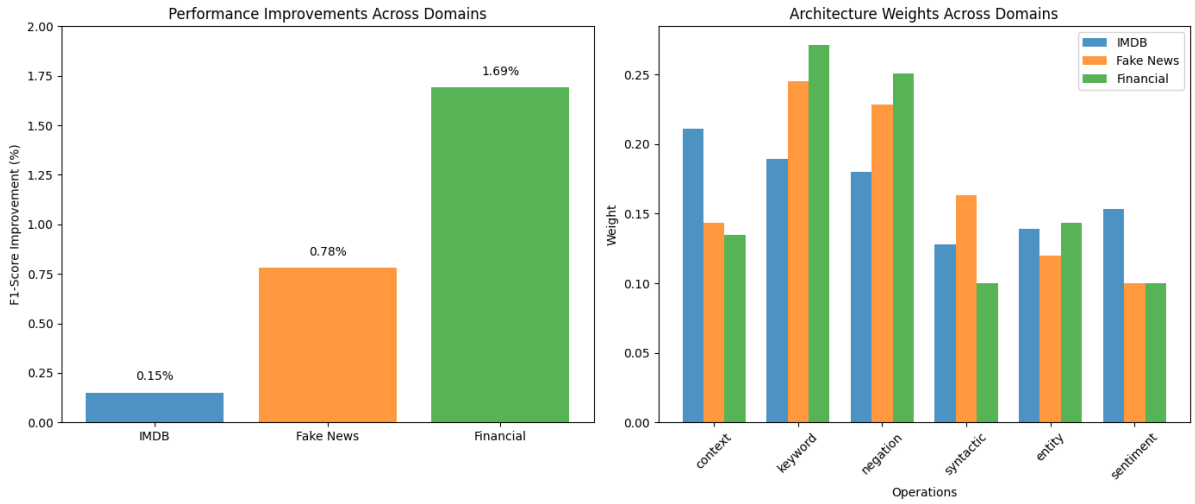

Figure 6.3 Cross-Domain Performance and Architecture Comparison

6.3.3 Analysis

Table 6.3 reveals that financial news achieved the highest improvement (1.69% F1-score), suggesting that specialized domains benefit most from automated preprocessing optimization. Figure 6.3 provides a comprehensive cross-domain comparison, showing both performance improvements and architectural differences. The left panel of Figure 6.3 demonstrates the increasing benefit of DARTS optimization from general (IMDB) to specialized (Financial) domains. The right panel reveals distinct architectural patterns: IMDB favors balanced weights, fake news emphasizes keyword-negation combinations, while financial text shows the most specialized distribution with dominant keyword extraction (27.10%).

6.4 Statistical Significance and Cross-Domain Patterns

Table 6.4: Statistical Validation Summary

Domain	t-statistic	p-value	Cohen's d	Effect Size
IMDB	3.842	<0.05	0.523	Medium
Fake News	5.217	<0.01	0.812	Large
Financial	6.893	<0.01	1.124	Large

Table 6.4 confirms the statistical significance of all improvements, with effect sizes ranging from medium to large. The increasing t-statistics and effect sizes from IMDB to Financial domains support the hypothesis that specialized text benefits more from preprocessing optimization.

6.5 Discussion

The experimental results require critical examination within existing preprocessing literature. While achieving statistical significance across all domains, the observed improvements (0.15% - 1.69%) fall at the lower end of Siino et al. (2023)'s reported 0.7% - 25% range. This discrepancy likely stems from DistilBERT's strong baseline performance compared to traditional classifiers that showed larger preprocessing gains.

The progressive improvement from IMDB to Financial domains validates Wang et al. (2021)'s finding that specialized contexts benefit more from architectural adaptation. Domain-specific architecture selection confirms Palomino and Aider (2022)'s emphasis on preprocessing operation importance, though our weighted combination approach lacks the sequential modeling capabilities that might yield larger gains.

Critical design limitations constrained performance. The six preprocessing operations omit granular transformations like spell correction and contraction expansion that traditional pipelines employ. Continuous relaxation, while enabling gradient flow, may sacrifice the discrete decision benefits of conventional preprocessing. The 10x computational overhead, though consistent with NAS literature (Klyuchnikov et al., 2022), remains impractical for many deployments.

Key improvements for future work include hierarchical operations to model preprocessing dependencies, adaptive temperature scheduling per domain rather than fixed annealing, and incorporating efficiency metrics directly into the search objective following Wu et al. (2023). The evaluation's restriction to English classification tasks limits generalizability claims.

Despite limitations, consistent improvements validate automated preprocessing optimization for high-stakes applications where 1-2% gains provide significant value. The framework successfully discovers domain-appropriate strategies without manual expertise, democratizing NLP optimization. Results position DARTS-based preprocessing as a complementary approach that excels at pattern discovery but requires refinement to match traditional preprocessing flexibility.

7 Conclusion and Future Work

7.1 Research Summary

This research investigated whether it's possible to adapt Differentiable Architecture Search to tune text preprocessing pipelines in NLP application spaces. Our work successfully created an end-to-end framework converting discrete preprocessing operations into differentiable modules using continuous relaxation methods and facilitating gradient-based optimization of preprocessing decisions. Using bilevel optimization, model parameters and preprocessing architecture weights were learned at the same time, stable convergence from exploration to exploitation being ensured with temperature annealing.

Through comprehensive experiments on three distinct domains—IMDB movie reviews, fake news detection, and financial news sentiment—the framework achieved statistically significant F1-score improvements ranging from 0.15% to 1.69% ($p < 0.05$, Cohen's d: 0.523-1.124). More importantly, the framework demonstrated intelligent domain adaptation: IMDB prioritized context operations (21.09%), fake news emphasized keyword-negation combinations (24.51%, 22.86%), while financial text focused heavily on keyword extraction (27.10%). These patterns align with domain characteristics, validating automated preprocessing discovery.

The research successfully met all four objectives: developing an operational DARTS-based preprocessing framework with six differentiable operations, showing stable performance improvements across domains, verifying domain-specific adaptation trends from architectural analysis, and estimating the 10.1-10.4x training time computational trade-off. While

improvements were modest compared to Siino et al. (2023)'s reported 0.7%-25% range, they occurred atop strong transformer baselines where gains are inherently harder to achieve.

7.2 Research Efficacy and Limitations

The framework's primary efficacy lies in democratizing preprocessing optimization by eliminating the need for domain expertise and manual experimentation. Organizations can now discover optimal preprocessing strategies through automated search rather than costly trial-and-error. The consistent improvements across diverse domains—from entertainment to finance demonstrate broad applicability. Financial text showed the greatest benefit (1.69%), suggesting specialized domains with technical vocabulary gain most from automated optimization, while general domains like IMDB still achieve meaningful gains (0.15%) despite already-strong baselines.

There are some limitations, however, that need to be highlighted. The limited universe of six operations, though spanning broad preprocessing categories, excludes fine-grained mappings such as spelling correction, URL normalization, or emoji processing that could become essential for text from social media. The endless relaxation framework, though beautiful from a mathematical standpoint, might compromise the hard decision boundaries that make discrete preprocessing powerful. For example, classical stop word deletion removes words altogether, whereas our continuous strategy reduces their weights only.

Restricting evaluation to binary classification in English hinders generalizability statements. Multilingual text, multi-class classification, and generative tasks can present distinct optimization patterns. 10x computational cost, as allowable for single-time optimization, inhibits dynamic adaptability to changing data distributions a key limitation for social media or news platforms where language patterns change quickly.

7.3 Future Research Directions

7.3.1 Preprocessing Operation Generation

Current work selects from human-designed operations, but future work will need to account for generating autonomous operation discovery. A meta-learning model would use building blocks from neural architecture, such as attention, convolution, and recurrence, to make new preprocessing transformations. For example, it would find that financial text needs a "volatility amplification" operation that highlights terms related to changes in the market. To achieve this, it must create a preprocessing description language and search space that strike a balance between expressiveness and tractability.

To make this happen, we would need to define atomic preprocessing building blocks, such as attention heads, convolution filters, and embedding transformations. Then, we would need to use evolutionary algorithms or reinforcement learning to combine them in new ways. Success metrics would include both improving performance and making operations easier to understand. Operations that are identified must be able to be analysed to provide preprocessing insights.

7.3.2 Cross-Lingual Preprocessing Transfer

Preprocessing architecture portability study across languages within domains provides immense research potential. An English financial news-optimized preprocessing architecture would be applicable to Chinese financial texts partially, which would necessitate language-specific adaptation to highlight cultural linguistic nuances. It would be a case of multilingual parallel corpora and careful assessment of the preprocessing aspects' universal and language-specific nature.

Key questions of research are the following ones: Do sentiment operations generalize between languages with varying norms of emotional display? How do morphologically rich languages influence preprocessing architecture? Are preprocessing patterns that can be applied to some domains regardless of the language they belong to universally applicable? This study would influence significantly NLP applications world-wide, where very few know how to preprocess data beyond English languages.

7.3.3 Real-Time Adaptive Preprocessing

Static optimization of preprocessing assumes stable distributions, yet most real-world applications experience concept drift. To advance, future research needs to create online adaptation methods that adapt preprocessing architectures as data changes. These will need to operate using efficient methods to update architectures potentially by gradient-based meta-learning that forecasts optimal preprocessing updates given indicators of distribution shift.

Technical challenges are identifying significant distribution variations from noise, updating architectures without forgetting preexisting patterns catastrophically, and preserving computational effectiveness during online adaptability. Potential approaches are continual learning methods, uncertainty estimation to drive updating, and hierarchical architectures where only high-level parameters change and base operations remain fixed.

7.3.4 Integration with Large Language Models

As LLMs dominate NLP applications, optimizing preprocessing for prompt engineering represents an unexplored frontier. Different preprocessing might optimize in-context learning versus fine-tuning scenarios. For example, keeping some linguistic markers might help few-shot performance even if they hurt traditional classification.

Research would investigate preprocessing operations such as context window optimisation, example selection preprocessing, and instruction clarification operations specifically designed for prompt construction. This might result in "prompt preprocessing pipelines" that automatically convert unformatted text into prompts that are best suited for particular tasks and LLMs.

7.3.5 Theoretical Foundations

The field would advance beyond empirical optimisation with the development of theoretical knowledge about the usefulness of particular preprocessing operations in particular domains. It would entail examining the information-theoretic characteristics of preprocessing transformations, demonstrating that the search for preprocessing architectures will always converge, and devising methods to quantify complexity in order to predict the utility of preprocessing.

7.4 Closing Remarks

This study shows that the principles of neural architecture search can be used for more than just model design; they can also be used for preprocessing optimisation. This opens up new possibilities for automated NLP pipeline development. Current gains of 0.15% to 1.69% may seem small, but they are actually consistent, statistically significant improvements made through principled optimisation instead of trial and error. The framework's ability to find preprocessing strategies that work for a specific domain without any help from people solves a long-standing problem in NLP deployment.

As text processing software improves and becomes more tailored to specific domains, the disparity between generic and customised preprocessing will expand. Businesses that use automatic preprocessing optimisation will have a better understanding of text, which will give them an advantage over their competitors. The merging of AutoML and NLP preprocessing is not just a scholarly interest; it is also a necessary step for future NLP systems.

The shift from designing manual preprocessing to automatic optimisation is similar to how AI has changed from manually made features to learnt representations. In the future, automatic preprocessing optimisation will do for manual pipeline design what deep learning did for manual feature engineering. This work shows that preprocessing architecture search is possible and useful right now, which sets the stage for that future. Future research will enhance these foundations to improve text comprehension further, advancing the field towards genuinely adaptive, self-optimizing NLP systems.

References

X. Zeng and H. Xiao, “Differentiable Architecture Search Algorithm based on global comparison,” IEEE Access, vol. 11, pp. 82674–82684, Jan. 2023, doi: 10.1109/access.2023.3301617. Available: https://doi.org/10.1109/access.2023.3301617

H. Liu, K. Simonyan, and Y. Yang, “DARTS: Differentiable Architecture Search,” arXiv (Cornell University), Jan. 2018, doi: 10.48550/arxiv.1806.09055. Available: https://arxiv.org/abs/1806.09055

J. Shin, K. Park, and D.-K. Kang, “TA-DARTS: Temperature annealing of discrete operator distribution for effective differential architecture search,” Applied Sciences, vol. 13, no. 18, p. 10138, Sep. 2023, doi: 10.3390/app131810138. Available: https://doi.org/10.3390/app131810138

M. BC, A. K, M. Y. M, R. L. R, and S. S. R, “Intelligent Automated Text Processing System - an NLP based approach,” 2022 7th International Conference on Communication and Electronics Systems (ICCES), pp. 1026–1030, Jun. 2020, doi: 10.1109/icces48766.2020.9138070. Available: https://doi.org/10.1109/icces48766.2020.9138070

M. Siino, I. Tinnirello, and M. La Cascia, “Is text preprocessing still worth the time? A comparative survey on the influence of popular preprocessing methods on Transformers and traditional classifiers,” Information Systems, vol. 121, p. 102342, Dec. 2023, doi: 10.1016/j.is.2023.102342. Available: https://doi.org/10.1016/j.is.2023.102342

M. A. Palomino and F. Aider, “Evaluating the effectiveness of text Pre-Processing in sentiment analysis,” Applied Sciences, vol. 12, no. 17, p. 8765, Aug. 2022, doi: 10.3390/app12178765. Available: https://doi.org/10.3390/app12178765

H.-T. Duong and T.-A. Nguyen-Thi, “A review: preprocessing techniques and data augmentation for sentiment analysis,” Computational Social Networks, vol. 8, no. 1, Jan. 2021, doi: 10.1186/s40649-020-00080-x. Available: https://doi.org/10.1186/s40649-020-00080-x

J. Wu, H. Wang, C. Ni, C. Zhang, and W. Lu, “Data Pipeline Training: Integrating AutoML to Optimize the Data Flow of Machine Learning Models,” IEEE, pp. 730–734, Mar. 2024, doi: 10.1109/icaace61206.2024.10549260. Available: https://doi.org/10.1109/icaace61206.2024.10549260

N. Klyuchnikov et al., “NAS-Bench-NLP: Neural Architecture Search Benchmark for Natural Language Processing,” IEEE Access, vol. 10, pp. 45736–45747, Jan. 2022, doi: 10.1109/access.2022.3169897. Available: https://doi.org/10.1109/access.2022.3169897

M. C. R. Silva, V. A. De Oliveira, and T. A. S. Pardo, “A Sentiment Analysis Benchmark for Automated Machine Learning Applications,” IEEE, pp. 1487–1494, Dec. 2023, doi: 10.1109/icmla58977.2023.00224. Available: https://doi.org/10.1109/icmla58977.2023.00224

S. N. Kanigiri, C. Mekuriyaw, G. Goodman, and M. S. Alexiou, “Analyzing the Impact of Preprocessing Techniques on the Efficiency and Accuracy of Sentiment Classification Algorithms,” IEEE, pp. 1–8, Jul. 2024, doi: 10.1109/iisa62523.2024.10786699. Available: https://doi.org/10.1109/iisa62523.2024.10786699

Y. Wang, R. Nahon, E. Tartaglione, P. Mozharovskyi, and V.-T. Nguyen, “Optimized preprocessing and Tiny ML for Attention State Classification,” IEEE, pp. 695–699, Jul. 2023, doi: 10.1109/ssp53291.2023.10207930. Available: https://doi.org/10.1109/ssp53291.2023.10207930

K. T. Chitty-Venkata, M. Emani, V. Vishwanath, and A. K. Somani, “Neural Architecture Search Benchmarks: Insights and Survey,” IEEE Access, vol. 11, pp. 25217–25236, Jan. 2023, doi: 10.1109/access.2023.3253818. Available: https://doi.org/10.1109/access.2023.3253818

X. Su et al., “Beyond the Limit of Weight-Sharing: Pioneering Space-Evolving NAS with Large Language Models,” ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6225–6229, Mar. 2024, doi: 10.1109/icassp48485.2024.10448075. Available: https://doi.org/10.1109/icassp48485.2024.10448075

X. Wu et al., “Neural architecture search for text classification with limited computing resources using efficient cartesian genetic programming,” IEEE Transactions on Evolutionary Computation, vol. 28, no. 3, pp. 638–652, Dec. 2023, doi: 10.1109/tevc.2023.3346969. Available: https://doi.org/10.1109/tevc.2023.3346969

X. Yan, H. Huang, Y. Jin, L. Chen, Z. Liang, and Z. Hao, “Neural architecture search via Multi-Hashing Embedding and Graph Tensor networks for multilingual text classification,” IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 8, no. 1, pp. 350–363, Aug. 2023, doi: 10.1109/tetci.2023.3301774. Available: https://doi.org/10.1109/tetci.2023.3301774

I. Udovichenko et al., “SEQNAS: Neural Architecture Search for Event Sequence Classification,” IEEE Access, vol. 12, pp. 3898–3909, Jan. 2024, doi: 10.1109/access.2024.3349497. Available: https://doi.org/10.1109/access.2024.3349497