INTERNATIONALE
HOCHSCHULE

IU International University of Applied Sciences
Course of Study: Applied Artificial Intelligence
Course: Project NLP - DLBAIPNLP01

Project report:

Project Natural Language processing

Task 1

written by

Laurenz Lattermann – 4239646

https://github.com/l-lattermann/tfidf-svm-sentiment-imdb.git

Contents

1 Introduction

1.1 Background and Motivation

Online platforms have become the primary medium for sharing public opinions about films. Movie reviews, in particular, offer a wealth of sentiment-rich text data. Automatically analyzing these reviews helps studios and platforms gauge public reception, improve marketing strategies, and guide content development. As Natural Language Processing (NLP) continues to evolve, sentiment analysis has become an increasingly important tool for extracting meaning from user-generated text. However, capturing subjective opinions reliably remains a complex task due to the informal and nuanced nature of human language.

1.2 Problem Statement

While sentiment analysis has made significant progress, many models still struggle with the intricacies of movie reviews. These reviews often contain slang, sarcasm, and mixed or ambiguous sentiments. Furthermore, domain-specific language and variations in writing style present challenges for general-purpose models. As a result, there is a need for an effective system that can process and classify sentiment in movie reviews using interpretable and computationally efficient methods.

1.3 Objectives of the Study

The main objective of this study is to develop a sentiment analysis pipeline for movie reviews by applying standard natural language processing techniques combined with machine learning classifiers. To achieve this, the project first involves collecting and preprocessing a labeled dataset of movie reviews from the publicly IMDB Large Movie Review Dataset. The preprocessing pipeline includes essential steps such as tokenization, stop word removal, normalization, and lemmatization, which are used to clean and prepare the raw text data. Following this, the processed text is transformed into a structured numerical format using Term Frequency–Inverse Document Frequency (TF-IDF) vectorization to extract informative features. A supervised learning model—specifically a Support Vector Machine (SVM)—is then trained on this encoded data. The model's performance is rigorously evaluated using standard classification metrics, including accuracy, precision, recall, and F1-score. Finally, the trained system is employed to classify unseen movie reviews, allowing it to determine the overall sentiment polarity expressed in new textual inputs.

1.4 Scope and Limitations

This study focuses on binary sentiment classification (positive vs. negative) in English-language movie reviews. It excludes multi-class classification (e.g., neutral sentiment), multilingual datasets, and deep learning approaches such as transformers. The system is developed and evaluated using primarily the IMDB dataset, and its transferability to other domains is not part of the project scope. Emphasis is placed on traditional machine learning to ensure interpretability and resource efficiency.

2 Background

2.1 Sentiment Analysis in NLP

Sentiment analysis, also known as opinion mining, is the field of study that analyzes people's opinions, attitudes, and emotions from text (Liu, 2012). Within NLP, sentiment analysis is typically framed as a text classification task where documents (such as movie reviews) are labeled according to the polarity of the expressed sentiment (positive vs. negative). Researchers have noted that sentiment analysis goes beyond mere topic detection, requiring an understanding of subjective language and context (Pang & Lee, 2008). Common applications include product review analysis, social media monitoring, and customer feedback evaluation. In the movie domain, sentiment analysis helps summarize critical reception and audience satisfaction. Over the years, this area has grown into a major subfield of NLP due to its practical importance and interesting linguistic challenges (Feldman, 2013). In summary, sentiment analysis in NLP provides automated methods to extract and quantify subjective information from text.

2.2 Machine Learning Approaches

A variety of machine learning approaches have been applied to sentiment analysis, evolving from early classification algorithms to advanced neural networks. Early work demonstrated that algorithms like Support Vector Machines can effectively classify sentiment in movie reviews (Pang, Lee, & Vaithyanathan, 2002). These models use features such as word frequencies or presence of indicative words (e.g., "great", "terrible") to learn the distinction between positive and negative sentiments. With the rise of deep learning, more complex models have been introduced – for example, convolutional and recurrent neural networks that capture word order and context have achieved impressive accuracy on sentiment tasks (Zhang, Wang, & Liu, 2018). Modern transformer-based models (like BERT) further push performance by leveraging pre-trained language understanding. While deep learning often yields higher accuracy due to its ability to learn nuanced language patterns, simpler algorithms remain competitive for moderate-sized datasets. In practice, sentiment analysis systems may employ a combination of techniques, selecting the approach based on available data and resources.

2.3 Bag-of-Words – Term Frequency – Inverse Document Frequency

The Bag-of-Words (BoW) model is a foundational technique in natural language processing in which a document is represented as a multiset of its words, disregarding grammar and word order but retaining multiplicity. Each document is transformed into a vector based on word frequency, resulting in a sparse and high-dimensional feature space (Jurafsky & Martin, 2021). While BoW captures basic lexical information, it does not account for the relative importance of words across documents. To address this limitation, the Term Frequency–Inverse Document Frequency (TF-IDF) weighting scheme is often applied. TF measures how frequently a term occurs in a document, while IDF reduces the weight of terms that are common across many documents, thereby emphasizing more informative and distinctive words.

2.4 Support Vector Machines (SVM)

Support Vector Machines (SVMs) are supervised learning algorithms used primarily for classification tasks, though they can also be applied to regression and outlier detection. The core principle of SVMs involves finding an optimal hyperplane that maximally separates data points of different classes in a high-dimensional space (Jurafsky & Martin, 2021). This separation is achieved by maximizing the margin between the nearest data points—known as support vectors—on either side of the decision boundary. For non-linearly separable data, the kernel trick is employed to project data into a higher-dimensional space where a linear separation becomes possible. Common kernels include linear, polynomial, and radial basis function (RBF) kernels. SVMs are known for their effectiveness in high-dimensional spaces and robustness against overfitting, particularly in cases where the number of features exceeds the number of samples (Hastie, Tibshirani, & Friedman, 2009). Their mathematical rigor and strong theoretical foundations make them a widely adopted method in various pattern recognition and machine learning tasks.

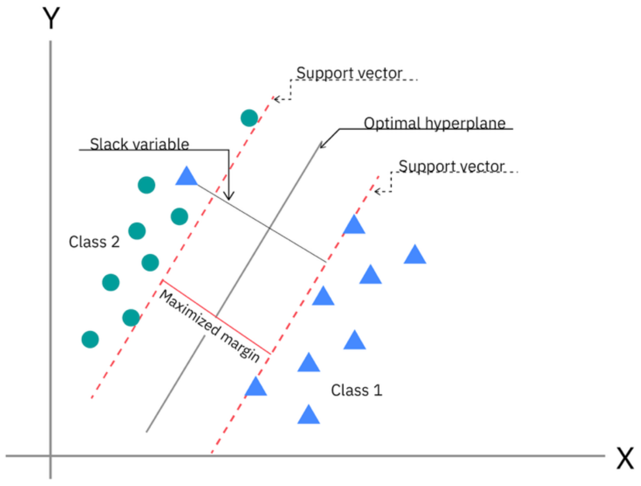

Figure 1: Illustration of a Support Vector Machine (SVM) classifier. IBM. (n.d.). Support vector machine. IBM. Retrieved July 29, 2025, from https://www.ibm.com/think/topics/support-vector-machine

3 Implementation Details

3.1 Project Structure and Modules

The project was organized into a modular and maintainable structure to support preprocessing, training, evaluation, and logging workflows. All core functionality was placed under the src/ directory, where submodules for configuration (config/), text preprocessing (preprocessing/), and model handling (svm/) were defined. The preprocessing module was structured to handle text normalization, lemmatization, and pipeline assembly, while model-related components were separated into training and prediction submodules. Configuration files such as training_params.yaml and logging templates were externalized in the config/ folder for reusability and clarity. Script-based entry points for model training and directory management were provided through training_pipeline.py. Supporting artifacts like logs, models, and documentation were stored in their corresponding top-level directories (logs/, models/, and README.md). Automated tests were placed in the tests/ directory and structured to mirror the source layout, ensuring each module was independently verifiable. Test logging and coverage reports were collected to evaluate the quality of the test suite.

3.2 Logging and Testing Strategy

During development, logging and testing strategies were implemented to ensure correct system behavior and simplify debugging. A logging framework using Python's built-in module was set up to record key events during data loading, preprocessing, model training, and evaluation—for instance, the number of reviews loaded or final performance metrics. Logs were saved to file to trace experiments and spot issues.

A basic testing strategy was also applied. Unit tests were written for key functions, such as HTML removal and sentence tokenization. Sanity checks ensured no reviews were empty and labels remained aligned. Quick end-to-end tests used small data subsets, while integration tests validated the full pipeline on sample data.

4 Methodology

At the start of the project, various modeling strategies were considered for sentiment classification, including transformer-based models like BERT, known for strong performance on NLP tasks (Devlin et al., 2019). However, BERT was deemed unsuitable, as it processes raw or minimally cleaned text using its own tokenizer (Sun et al., 2019), leaving no room to evaluate custom preprocessing. Since the project's focus was on assessing the effects of different preprocessing steps, a classical approach using TF-IDF vectorization and a Support Vector Machine (SVM) was chosen. This setup allows transparent tuning, aligns with the educational goals, and offers competitive performance.

4.1 The Data set

For this study, only the *"Stanford Large Movie Review Dataset v1.0"* was used, as alternative sources—such as web scraping or open-source review platforms—were excluded due to legal restrictions or lack of publicly available review text. The dataset contains 50,000 movie reviews equally divided into positive and negative sentiments and includes additional unlabeled data, making it well suited for supervised sentiment classification tasks (Maas et al., 2011; Stanford AI Lab, 2011). At the beginning of the project, a reduced subset of reviews was utilized for initial testing and pipeline validation. In the final phase, the complete dataset was employed for training and evaluation. The data was split into separate training and test sets, and measures were taken to prevent any sample overlap, thereby maintaining the integrity of the evaluation process.

4.2 Preprocessing Pipeline

To improve the quality of the input data for modeling, raw text reviews were subjected to a standardized preprocessing pipeline. This pipeline incorporated established natural language processing techniques as described by Manning, Raghavan, and Schütze (2008). Each step was designed to reduce noise, normalize linguistic variation, and prepare the data for feature extraction. The key stages of this pipeline are outlined below.

4.2.1 Text Cleaning

The text was first normalized using a multi-step cleaning pipeline. All characters were converted to lowercase, and HTML tags—which are frequently present in the dataset due to formatting within user reviews—were removed. Incorrect quotation marks and URLs were also eliminated. Lastly, non-alphanumeric characters and unnecessary whitespace were removed. These steps ensured a clean and uniform text format for further processing.

4.2.2 Normalization

In the normalization stage of preprocessing, contractions and informal language were systematically expanded and standardized. Common contractions such as *“don’t”* or *“can’t”* were replaced with their full forms (e.g., *“do not”*, *“cannot”*) to ensure grammatical consistency across the dataset. Additionally, internet slang and informal expressions—frequently found in user-generated content—were substituted with their formal equivalents. Examples include the replacement of *“lol”* with *“funny”* and *“brb”* with *“be right back”*. These transformations were applied using predefined substitution dictionaries and regular expressions, enhancing lexical clarity and preparing the text for more effective downstream token-based analysis.

4.2.3 Spelling correction

Various methods for automatic spelling correction were tested, including rule-based approaches with TextBlob as well as transformer-based models. However, both strategies yielded poor results, often introducing more mistakes than eliminating. Instead of relying on explicit correction, misspelled words were implicitly filtered out by the *“max_features”* parameter of the *TfidfVectorizer*, which limits the feature space to the most frequent terms and excludes rare tokens such as typos (scikit-learn developers, 2024)

This approach avoids unnecessary dimensionality growth and results in only minimal information loss, as misspellings are generally infrequent and thus unlikely to distort the overall feature representation in a well-balanced dataset.

4.2.4 Lemmatization

To reduce morphological variation, lemmatization was applied. The text was processed using the language model *“en_core_web_sm”*. The *“en_core_web_sm”*. This model from spaCy is a light-weight English pipeline that includes part-of-speech tagging, dependency parsing, and named entity recognition, making it suitable for many standard NLP tasks (Explosion AI, 2023). Each review was tokenized and analyzed syntactically, and the base (lemma) of each word was extracted. This allowed inflected forms—such as *“running”*, *“ran”*, or *“runs”*—to be reduced to their canonical form *“run”*. By applying this transformation, lexical variation was minimized, enabling more consistent feature representation. The lemmatized tokens were then joined back into normalized strings and passed on to the next preprocessing stage.

4.2.5 Stop Word Removal

Stop words—commonly used terms such as “the”, “and”, or “is”—were automatically removed during feature extraction, as this functionality is built into the *TfidfVectorizer* from *scikit-learn*. Eliminating these high-frequency, low-information words reduces dimensionality and allows the model to focus on more meaningful features relevant to sentiment classification (scikit-learn developers, 2024).

4.3 Encoding

The preprocessed reviews were transformed into numerical feature vectors using a TF-IDF vectorizer, with all hyperparameters specified externally via a YAML configuration file to enable systematic tuning. The parameter *“max_features”* was used to restrict the vocabulary to the most frequent terms, thereby reducing dimensionality. The *“ngram_range”* defined whether unigrams, bigrams, or trigrams were included. Common stop words were removed using the *“stop_words='english'”* setting, and all tokens were lowercased via the *“lowercase=True”* parameter. Further refinements were introduced through *“use_idf”* and *“smooth_idf”*, which influenced the calculation of inverse document frequencies, and *“sublinear_tf”*, which applied logarithmic scaling to term frequencies (scikit-learn developers, 2024).

In total, 72 distinct combinations of these vectorization parameters were defined and applied, resulting in 72 differently encoded datasets. This allowed for a comparative evaluation of all variants, with the goal of identifying the top-performing vectorizer configuration for downstream model training.

4.4 Model Selection and Training

4.4.1 Hyperparameter Tuning (GridSearch)

To optimize the classification models, a grid search was conducted using scikit-learn’s *GridSearchCV*. Different combinations of hyperparameters for each model were systematically explored by training and validating across multiple parameter grids (Apppendix A.). Every encoded dataset was then passed to a *LinearSVC* model, for which classifier-level hyperparameters—specifically *C, tol, max_iter*, and *class_weight*—were fine-tuned using a nested grid search. Three-fold cross-validation was applied during the tuning process to ensure statistical reliability and mitigate overfitting. As the primary evaluation criterion, the F1-score was selected to guide model selection, as it balances precision and recall and remains robust under mild class imbalance. Through this structured and exhaustive search, optimal model configurations were identified in a reproducible and systematic manner.

4.4.2 Evaluation Metrics

To evaluate the performance of the sentiment analysis model, a comprehensive pipeline was implemented, integrating *"TF-IDF"-based* feature extraction, extensive *"GridSearchCV"*-driven optimization, and standard classification metrics. A linear *"Support Vector Machine (SVM)"* served as the core classifier and was trained on 40,000 labeled movie reviews, while a separate holdout set of 10,000 samples was used for testing. Feature encoding was conducted using a *"TF-IDF vectorizer"*, with the parameter *"max_features"* varied across 80,000, 50,000, and 30,000; *"ngram_range"* set to (1,2), (1,3), and (2,3); and binary configurations of *"use_idf"*, *"smooth_idf"*, and *"sublinear_tf"*. All vectorizations applied English stopword removal and lowercasing via the *"stop_words"* and *"lowercase"* parameters, respectively. Each resulting feature matrix was paired with an *"SVM"* trained through grid search over 72 hyperparameter combinations involving *"C"* ∈ {0.5, 1, 2}, *"tol"* ∈ {1e−5, 1e−4, 1e−3}, *"max_iter"* ∈ {1000, 2000, 4000}, and toggling of *"class_weight='balanced'"*. This configuration enabled fine-grained optimization tailored to each vectorization setting. Model performance was assessed using metrics outlined by Sokolova and Lapalme (2009), with *"accuracy"* employed as a baseline indicator of correctness across the balanced dataset. In anticipation of potential asymmetry in class behavior, additional evaluations were conducted using *"precision"*, *"recall"*, and *"F1-score"* for each class. The metric *"precision"* quantified the proportion of true positives among all positive predictions, while *"recall"* measured the ability to retrieve all true positives. The *"F1-score"*, representing the harmonic mean of both, served as the principal criterion for model selection due to its resilience against imbalanced errors. A *"confusion matrix"* was also generated per evaluation to facilitate visual inspection of misclassification patterns. Through this multifactor evaluation strategy, model selection was guided by both overall accuracy and balanced class-wise performance.

5 Results and Evaluation

The sentiment classification results revealed that multiple configurations achieved high performance, with F1-scores consistently ranging between 0.83 and 0.90. The best-performing setups reached an F1-score of 0.90 and an overall accuracy of 90%, showing strong precision and recall balance for both positive and negative sentiment classes.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
neg	0.91	0.88	0.90	4968
pos	0.89	0.91	0.90	5032
accuracy	–	–	0.90	10000
macro avg	0.90	0.90	0.90	10000
weighted avg	0.90	0.90	0.90	10000

Table 1: Evaluation Metrics of the best SVM classiefier

These outcomes were typically associated with C = 0.5, class_weight = 'balanced', a low tolerance (1e-05), and sufficient iterations (max_iter = 1000). The use of class balancing proved crucial, especially under slightly imbalanced datasets, by preventing the model from biasing toward the majority class. Performance decreased noticeably when class_weight was set to None, suggesting that the model had more difficulty capturing the minority class, leading to slightly lower recall and hence reduced F1-scores (typically 0.89). Similarly, increasing C to 1 did not yield improvements and sometimes worsened performance due to reduced regularization, potentially leading to mild overfitting and instability in generalization. A more significant drop occurred in models with F1-scores of 0.83–0.85.

These setups often used higher *“tolerance”* values or lacked *“class_weight”* balancing, but more critically, they suffered from limited expressive power due to suboptimal feature extraction (e.g., fewer features or reduced *“ngram_range”* coverage). Excluding unigrams in favor of only bigrams/trigrams weakened the model’s ability to detect simple sentiment markers like “great” or “bad.” A reduced feature space—e.g., via low *“max_features”*—further limited access to informative tokens and hindered detection of nuanced sentiment. Overall, while tuning parameters such as *“tol”*, *“C”*, and *“class_weight”* impacted robustness, the most decisive factors were adequate regularization, rich *“ngram_range”* coverage including unigrams, and balanced *“class_weight”*. Models with these traits consistently performed best. The optimal parameters for the vectorizer and for classifier training are listed below.

Parameter	Value
max_features	80000
ngram_range	(1,2)
stop_words	“english”
lowercase	True
use_idf	True
smooth_idf	True
sublinear_tf	True

Parameter	Value
C	0.5
class_weight	balanced
max_iter	1000
Tol	1e-05

Table 2: Best parameters for the Vectorizer, Table 3: Best parameters for the SVM classifier

6 Discussion

The sentiment classification model achieved strong results on the IMDb dataset, with an F1-score of 0.90, accuracy of 90%, and balanced precision and recall across both sentiment classes. These outcomes confirm the effectiveness of traditional machine learning pipelines using TF-IDF and linear SVMs when appropriately tuned.

Key hyperparameters contributing to high performance included C = 0.5, class_weight = 'balanced', and a low tol = 1e-05. Models without class balancing or with higher C values typically suffered reduced recall on the minority class and a drop in F1 to around 0.89. Subpar results (F1 ≈ 0.83–0.85) were often linked to limited feature expressiveness, such as reduced “*n-gram*” ranges or smaller “*max_features*”.

Compared to prior work, the model’s performance is competitive, as shown in the table below.

Study / Authors	**Model**	**Accuracy**
Al Hosani et al. (2024)	SVM + TF-IDF (Spark MLlib)	88.75%
Farasalsabila et al. (2023)	SVM + BoW	88.59%
Farasalsabila et al. (2023)	SVM + TF-IDF	91.27%
Talibzade (2023)	SVM + TF-IDF	90.00%
Talibzade (2023)	BERT	98.00%
This Study (2025)	**SVM + TF-IDF**	**~ 90%**

Table 4: Comparative performance of SVM- and BERT-based sentiment classification models on the IMDb dataset (English, binary classification). The table summarizes accuracy results from this study and three peer-reviewed publications—Al Hosani et al. (2024), Farasalsabila et al. (2023), and Talibzade (2023).

Accuracy is used as a metric for comparison here, because F1-Score is not consitently reported in the studies found. The model developed in this study achieved a competitive accuracy of approximately 90%, aligning closely with the performance reported by Talibzade (2023) and exceeding that of Al Hosani et al. (2024) and Farasalsabila et al. (2023) using BoW. While slightly below the peak performance of Farasalsabila et al.’s TF-IDF model (91.27%) and the transformer-based BERT approach (98%). This SVM + TF-IDF implementation demonstrates that traditional machine learning methods remain robust and efficient for sentiment classification, especially when considering computational cost and simplicity.

6.1 Lessons Learned

Rich feature representations, including higher n-gram ranges and larger vocabularies (max_features), significantly contributed to model generalization. Linear SVMs, when paired with well-engineered features, continue to perform competitively even against more complex alternatives. Hyperparameter optimization (e.g., tuning C, tol) led to only marginal performance gains, suggesting that vectorization and feature design had a greater impact than fine-tuning the classifier itself.

6.2 Limitations

This study was limited to binary sentiment classification using classical machine learning methods. Deep learning models such as BERT or LSTM were not explored, although they have demonstrated superior performance in sentiment analysis tasks, particularly when handling complex linguistic structures or subtle contextual cues (Devlin et al., 2019).

Furthermore, no detailed error analysis was conducted to investigate specific misclassification patterns. The model’s ability to generalize to other domains, such as social media or customer service data, remains untested and is a potential area for future research.

References

Al Hosani, S. A., Hassooni, A. M., Al Jarwan, M. M., & Abul, O. (2024). Sentiment analysis of movie reviews using Apache Spark MLlib: A big data approach. In 2024 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT) (pp. 282–285). IEEE. https://doi.org/10.1109/BDCAT63179.2024.00052

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 1, 4171–4186. https://doi.org/10.18653/v1/N19-1423

Explosion AI. (2023). spaCy models documentation. https://spacy.io/models/en

Farasalsabila, F., Lestari, V. B., Cahyo, D. D. N., Hanafi, H., Lestari, T., Al Islami, F. R., & Maulana, M. A. (2023). Sentiment analysis for IMDb movie review using Support Vector Machine (SVM) method. *Inform: Jurnal Ilmiah Bidang Teknologi Informasi dan Komunikasi, 8*(2), 90–94. https://doi.org/10.25139/inform.v8i2.5700

Feldman, R. (2013). Techniques and applications for sentiment analysis. Communications of the ACM, 56(4), 82–89. DOI: 10.1145/2436256.2436274

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction (2nd ed.). Springer. https://doi.org/10.1007/978-0-387-84858-7

Jurafsky, D., & Martin, J. H. (2021). Speech and Language Processing (3rd ed., draft). Stanford University. https://web.stanford.edu/~jurafsky/slp3/

Liu, B. (2012). Sentiment Analysis and Opinion Mining. Morgan & Claypool Publishers. (Synthesis Lectures on Human Language Technologies, Vol. 5, No. 1).

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (pp. 142–150). Association for Computational Linguistics.

Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval, 2(1–2), 1–135. https://doi.org/10.1561/1500000011

Pang, B., Lee, L., & Vaithyanathan, S. (2002). *Thumbs up? Sentiment classification using machine learning techniques*. Proceedings of EMNLP 2002, 79–86.

scikit-learn developers. (2024). TfidfVectorizer — scikit-learn 1.x documentation. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Information Processing & Management, 45(4), 427–437. DOI: 10.1016/j.ipm.2009.03.002

Stanford AI Lab. (2011). Large Movie Review Dataset v1.0 [Data set]. https://ai.stanford.edu/~amaas/data/sentiment/

Sun, C., Qiu, X., Xu, Y., & Huang, X. (2019). How to fine-tune BERT for text classification? China National Conference on Chinese Computational Linguistics, 194–206. https://doi.org/10.1007/978-3-030-32381-3_16

Talibzade, R. (2023). *Sentiment analysis of IMDb movie reviews using traditional machine learning techniques and transformers* [Unpublished term paper]. ADA University. https://doi.org/10.13140/RG.2.2.29464.16644

Zhang, L., Wang, S., & Liu, B. (2018). *Deep learning for sentiment analysis: A survey*. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8(4), e1253. https://doi.org/10.1002/widm.1253

Appendices

A. Model Hyperparameters

Vectorizer Hyperparameters

Parameter	Values	Description
`max_features`	[80000, 50000, 30000]	Limits the number of features (vocabulary size) to the top N most frequent terms.
`ngram_range`	[(1, 2), (1, 3), (2, 3)]	Defines the range of n-gram sizes to include, e.g., unigrams to trigrams.
`stop_words`	['english']	Removes common English stopwords (e.g., "the", "is") from the text.
`lowercase`	[True]	Converts all text to lowercase before tokenization.
`use_idf`	[True, False]	Whether to enable inverse document frequency (IDF) reweighting.
`smooth_idf`	[True, False]	Adds 1 to document frequencies to prevent division by zero in IDF computation.
`sublinear_tf`	[True, False]	Applies logarithmic scaling to term frequency to dampen the effect of large counts.

Vectorizer Hyperparameters

Parameter	Values	Description
`C`	[0.5, 1, 2]	Regularization parameter: smaller values specify stronger regularization.
`tol`	[1.0e-5, 1.0e-4, 1.0e-3]	Tolerance for stopping criteria; smaller values mean more precise convergence.
`max_iter`	[1000, 2000, 3000, 4000]	Maximum number of iterations for optimization solver.
`class_weight`	[None, 'balanced']	Adjusts weights inversely proportional to class frequencies for imbalanced data.

B. GitHub Repository Link

https://github.com/l-lattermann/tfidf-svm-sentiment-imdb.git