Report title

Giovanni Orlandi
Politecnico di Torino
Student id: s355320
s355320@studenti.polito.it

***Abstract*—This work addresses supervised news article classification problem using TF–IDF-based textual representations. A multi-stage hyperparameter optimization strategy identifies Linear SVM as the most effective approach. The final model achieves a macro F1-score of 0.736 on the public leaderboard, substantially outperforming a naive baseline, while class imbalance and semantic overlap remain the main limiting factors.**

I. Problem overview

This competition introduces the task of automatically classifying news articles into thematic categories. In particular, the goal is to assign each article to one of the following seven classes: International News (0), Business (1), Technology (2), Entertainment (3), Sports (4), General News (5), and Health (6).

The dataset is split into two subsets:

- a *development* set, containing 79,997 labeled news articles
- an *evaluation* set, comprising 20,000 unlabeled articles

The development set is employed for model training and validation, after which the trained models are used to produce predictions on the evaluation set. The use of external datasets is explicitly forbidden.

A preliminary inspection of the development data reveals a noticeable class imbalance. As shown in Figure 1, class 0 (International News) accounts for nearly 30% of the observations, while several other classes are significantly underrepresented. This class imbalance introduces an intrinsic asymmetry in the classification task, potentially biasing model evaluation toward dominant categories [1].

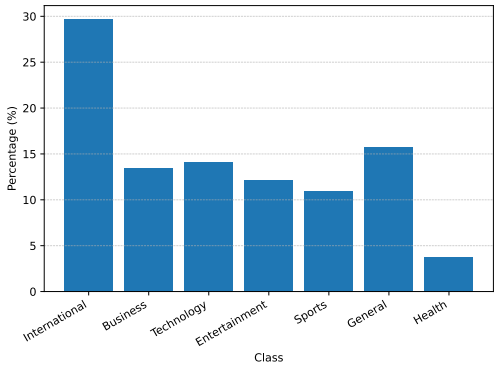

Fig. 1: Distribution of target classes in the development set.

The dataset includes five input features (excluding the *identifier*), whose characteristics are briefly analyzed in what follows.

A. Source

The *source* feature represents the news outlet or publisher as a categorical variable. A naive one-hot encoding of this feature would result in a very high-dimensional representation, as it includes 1,358 unique values. However, approximately 86.7% of the articles belong to the 50 most frequent sources, suggesting that category grouping strategies may be effective.

Additionally, around 0.3% of the entries correspond to missing values, encoded as a specific placeholder `(\N)`.

B. Title and article

The *title* and *article* fields contain the main textual content of each news item. Previous studies indicate that textual features are among the most informative sources for news classification tasks [2]. At the same time, they require substantial preprocessing, as the raw text often includes HTML artifacts and noisy tokens, likely originating from web scraping procedures. Consequently, appropriate text cleaning and representation techniques are essential to extract meaningful information from these fields. In addition, the length and structure of textual content vary substantially across articles, contributing to highly sparse and heterogeneous feature representations [3].

C. Page rank

The *page_rank* feature is an ordinal variable taking integer values between 2 and 5. Its distribution is highly skewed, with approximately 92% of the articles having a page rank equal to 5. Moreover, for lower page rank values (2 and 3), the associated articles belong exclusively to the Technology class. This observation indicates a highly class-conditional distribution of the page rank feature, which motivates a careful treatment during preprocessing.

D. Timestamp

The *timestamp* feature encodes the publication date and time of each article. Missing values are represented by the placeholder string `0000-00-00 00:00:00` and account for approximately 33% of the dataset. No clear class-dependent pattern was observed for missing timestamps when comparing class-wise distributions. For valid entries, publication dates span from mid-2004 to early 2008. A noticeable gap of approximately one and a half years is

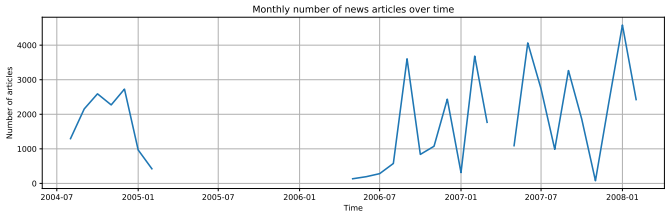

Fig. 2: Monthly number of news articles over time.

present in the middle of this interval, as shown in Figure 2. These characteristics introduce noise into the signal and make the temporal information difficult to analyze effectively.

The insights obtained from this analysis informed the subsequent preprocessing choices.

II. Proposed approach

A. Preprocessing

Given the unstructured nature of textual data, preprocessing represents a crucial step of the classification pipeline [4]. Before the feature-specific preprocessing procedures, duplicate records were identified and removed from the dataset. This step was performed to avoid conflicting labels for identical or near-identical samples, which are known to negatively affect the learning process in supervised classification tasks [5]. Duplicates were handled according to the following rules:

- exact duplicates were collapsed into a single record, discarding redundant copies
- inconsistent records sharing identical features but associated with different labels were removed
- records differing only in the *timestamp* were merged by retaining the earliest timestamp and assigning the label by majority voting
- records differing only in the *page rank* were merged by assigning the label by majority voting and the page rank by mode
- records differing only in the *source* were merged by assigning the label by majority voting and the source by mode

More aggressive deduplication strategies, including fuzzy comparisons based on the textual fields (*title* and *article*), were also evaluated, but led to degraded validation performance and were therefore discarded.

The preprocessing steps applied to each feature are described below.

1) Source: The *source* feature was treated as a categorical variable. Missing values were explicitly handled by introducing a dedicated category (`MISSING`), allowing the model to distinguish between unknown and observed publishers.

Given the high cardinality of this feature (1,358 unique values), different encoding strategies were evaluated. A naive one-hot encoding results in a high-dimensional representation. However, in this task, the overall feature space is already dominated by TF–IDF textual features; as a consequence, the relative contribution of the source encoding to the total dimensionality is limited. This approach was empirically compared with a truncated version retaining only the 50 most frequent sources. Experimental results showed that the full one-hot encoding consistently achieved better validation performance so it was the one selected for the final pipeline.

2) Page rank: Page rank values 2 and 3 were merged into a single category. This choice was motivated by the fact that all observations associated with these values belong exclusively to the Technology class.

Missing values were encoded using a dedicated value set to zero.

3) Timestamp: Timestamp preprocessing consisted of extracting multiple temporal features from the original timestamp. In particular, for hour, month and day of the week, two alternative encodings were evaluated: raw integer values and cyclical encodings using sine and cosine transformations [6]. Empirical results showed that raw values consistently led to better validation performance and were therefore preferred.

Additional temporal features, such as a weekend indicator, were also evaluated but did not provide further improvements. The final set of retained temporal features includes the year, month, day of the week and hour, all represented as raw numerical values.

Missing timestamps were handled by assigning a dedicated value of zero to all extracted temporal features and by introducing an additional boolean indicator flag.

4) Title and article: Both the *title* and *article* fields consist of unstructured textual data, for which several representation approaches have been proposed in the literature [4]. Given the effort required to properly implement multiple text representation methods, a single, well-established approach was adopted, namely TF–IDF [7]. This approach is a term-weighting scheme that represents documents as vectors. It combines term frequency with inverse document frequency, emphasizing terms that are frequent within a document but rare across the corpus.

Before text encoding, a dedicated cleaning procedure was applied to mitigate noise present in the raw data.

1) HTML entities were unescaped and all text was lowercased.
2) Frequent HTML-related noise tokens (`http`, `https`, `www`, `com`) were removed.
3) Numerical expressions were replaced with category-specific tokens using regular expressions; residual numeric patterns were discarded.
4) For the *article* field only, text fragments shorter than five characters were removed.

This lightweight cleaning strategy was empirically found to outperform more aggressive text normalization approaches.

After preprocessing, both textual fields were encoded using TF–IDF representations with English stopwords removed. Different configurations of TF–IDF hyperparameters were evaluated. For the *article* field, both word-level and character-level representations were considered. The final parameter settings and corresponding results are described Section II-C.

A dimensionality reduction step was evaluated; however, in line with existing literature [3], [8], models capable of directly handling sparse high-dimensional representations achieved superior performance III.

For these reasons, no explicit dimensionality reduction was ultimately applied. Instead, sparsity was controlled through tuning TF–IDF parameters (e.g., `min_df` and `max_df`), and all features were processed using sparse matrix representations.

In addition to text encoding, two metadata features were included: *article length* and *title length*, used as coarse proxies for document structure and verbosity.

B. Model selection

The following classification algorithms were evaluated:

- **Naive Bayes**: widely used as an efficient baseline for text classification due to its scalability and robustness to high-dimensional sparse representations such as TF–IDF [9].
- **XGBoost**: applied to text classification to model non-linear feature interactions, typically in combination with dimensionality reduction techniques [4], [10].
- **Linear SVM**: among the most effective methods for high-dimensional text classification, consistently achieving strong performance with sparse representations such as TF–IDF [3].
- **Ridge Classifier**: linear classifiers with L2 regularization to control model complexity, successfully applied to high-dimensional text classification tasks [11].
- **Logistic Regression**: standard discriminative model for document and news classification, providing competitive performance and well-calibrated probabilistic outputs on sparse textual data [12].
- **Stochastic Gradient Descent (SGD)**: stochastic optimization framework for training linear models, enabling scalable learning of SVM- and logistic regression–style objectives on high-dimensional sparse text data [13].

Given the high computational cost associated with exhaustive hyperparameter tuning, a multi-stage model selection strategy was adopted, as described in the following section.

C. Hyperparameter tuning

Two main groups of hyperparameters were considered:

- preprocessing-related parameters
- model-specific parameters

Hyperparameter search was conducted using the Tree-structured Parzen Estimator (TPE) algorithm [14], as implemented in the Optuna Python library [15]. This Bayesian optimization method allows for a more efficient exploration of the hyperparameter space compared to other approaches.

The development set was split into:

- 80% train-validation set, used for hyperparameter optimization via Stratified K-Fold cross-validation within each Optuna trial
- 20% test set, used for final model evaluation

TABLE I: TF–IDF configurations adopted for textual features.

	include	ngram	min_df	max_df	norm	sub_tf
title	yes	(1,2)	2	0.8	l2	True
article - word	yes	(1,3)	6	0.9	l2	True
article - char	yes	(3,5)	5	0.9	l2	True
svd	no	/	/	/	/	/

Due to space constraints, the full hyperparameter search spaces are not reported here and are instead provided in the project repository [1].

Although preprocessing and model hyperparameters are not strictly independent, they were optimized separately as a practical approximation, in order to keep the computational cost of hyperparameter optimization feasible.

Preprocessing parameters included:

- TF–IDF configuration parameters
- inclusion of the title field
- character-level TF–IDF encoding for the article text
- Truncated SVD-based dimensionality reduction

For this stage, Optuna was configured with 30 trials and a Stratified K-Fold cross-validation with $k = 2$.

Results indicated that the preprocessing configuration reported in Table I consistently yielded the best performance.

Model hyperparameters were subsequently optimized using a multi-stage strategy. This approach allowed us to explore a wide range of model configurations while keeping the computational cost under control:

1) an initial coarse hyperparameter search (10 trials) using stratified k-fold cross-validation ($k = 2$) to identify promising models, evaluated using internal validation metrics
2) a refined hyperparameter search (40 trials) on the top two models using stratified k-fold cross-validation ($k = 3$), with model selection based on internal metrics and external monitoring on the public leaderboard
3) a post-hoc evaluation of alternative class-weighting schemes on the selected model to mitigate class imbalance and improve performance on under-represented classes
4) a final extensive hyperparameter optimization (100 trials) on the best-performing model using stratified k-fold cross-validation ($k = 3$)

The public leaderboard was used solely for qualitative monitoring and sanity checks, and did not influence any model or hyperparameter selection decisions.

III. Results

Empirical results from the first stage indicate that Linear SVM and SGD consistently outperformed the other evaluated approaches. XGBoost was also considered in combination with dimensionality reduction via truncated SVD; however, this configuration yielded inferior performance compared to the top-performing linear models and was therefore not pursued further.

[1] search_spaces.py

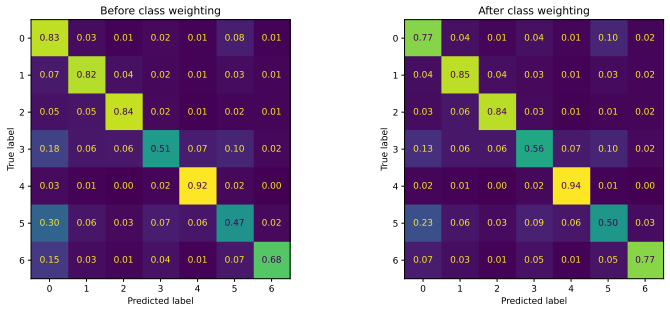

Fig. 3: Row-normalized confusion matrices illustrating the effect of class weighting.

In the second stage, Linear SVM emerged as the best-performing model, achieving an internal macro F1-score of 0.7233 (0.729 on the public leaderboard), marginally outperforming SGD, which obtained an internal macro F1-score of 0.7214 (0.727 on the public leaderboard). For this reason, Linear SVM was selected for subsequent stages and, given the close similarity between their underlying optimization objectives, further evaluation of both models was deemed redundant.

A detailed analysis of the prediction outputs revealed systematic confusion among specific classes. In particular, substantial confusion was observed among classes 0, 3, and 5, along with consistently poor recall for class 6, as shown in the left panel of Figure 3. The former issue is likely attributable to intrinsic semantic overlap between the corresponding categories (International News, Entertainment, and General News), while the latter is plausibly explained by the strong class imbalance present in the dataset.

These observations motivated the third optimization stage, evaluating alternative class-weighting schemes. Starting from the best Linear SVM configuration from Stage 2, the optimal setting corresponded to weights inversely proportional to the empirical class frequencies: $\{0 : 1.0, 1 : 2.2, 2 : 2.1, 3 : 2.4, 4 : 2.7, 5 : 1.8, 6 : 7.6\}$.

The model with the new class-weight configuration achieved an internal macro F1-score of 0.728 (0.732 on the public leaderboard), driven by improved recall on under-represented classes at the expense of a mild performance decrease for dominant categories, as shown in the right panel of Figure 3.

The final stage consisted of an extensive hyperparameter optimization of the selected Linear SVM model.

The final hyperparameter configuration adopted for the submission is reported below:

- C: *0.04773295631615594*
- maximum number of iterations: *5000*
- class weights: $\{0 : 1.0, 1 : 2.2, 2 : 2.1, 3 : 2.4, 4 : 2.7, 5 : 1.8, 6 : 7.6\}$
- dual: *True*

This configuration achieved a macro F1-score of 0.7333 on the test set and, after retraining on the full development dataset, obtained a score of 0.736 on the public leaderboard. The progressive performance improvements across the different optimization stages are summarized in Table II.

		Internal F1	Public F1
Stage 1	SVM	0.719	x
	SGD	0.7201	x
Stage 2	SVM	0.7233	0.729
	SGD	0.7214	0.727
Stage 3	SVM	0.728	0.732
Stage 4	SVM	0.733	**0.736**
Baseline		x	0.443

TABLE II: Performance across optimization stages.

IV. Discussion

The proposed approach substantially outperforms the naive baseline reported on the public leaderboard (as shown in Table II) and achieves competitive performance relative to other submitted solutions. Within the evaluated setting, Linear SVM emerged as the best-performing model. Overall, the results suggest that linear models operating on high-dimensional sparse TF–IDF representations are more effective than the more complex alternatives considered in this study.

Nevertheless, the semantic overlap among classes 0, 3, and 5 emerges as a major bottleneck for further performance improvements. Similar limitations induced by class overlap have been widely reported in the literature [16]. As reflected in the confusion matrices, misclassifications are predominantly concentrated among these classes and persist across different models and hyperparameter configurations. This behavior suggests that the limitation is primarily driven by the dataset characteristics and the lexical nature of TF–IDF representations. Consequently, the performance achieved by the proposed pipeline can be considered close to the upper bound attainable within this modeling framework.

Several directions may be considered to further improve the obtained results:

- Techniques specifically designed to address class overlap, such as multi-stage classification strategies [17] or more aggressive feature selection methods [18], could be explored more extensively.
- A more substantial improvement is likely to require a fundamentally different representation strategy. In particular, dense semantic embeddings, such as Word2Vec-based [19] representations or transformer-based encodings [19], [20], may better capture contextual and semantic information. Although these approaches were not investigated in this work in order to maintain a focused analysis of TF–IDF-based models, existing literature suggests that they represent a promising direction for future research [21].

References

[1] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[2] F. Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys*, vol. 34, no. 1, pp. 1–47, 2002.

[3] T. Joachims, "Text categorization with support vector machines," in *European Conference on Machine Learning*, 1998.

[4] K. Kowsari *et al.*, "Text classification algorithms: A survey," *Information*, 2019.

[5] B. Frénay and M. Verleysen, "Classification in the presence of label noise: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 845–869, 2014.

[6] T. Mahajan, "An experimental assessment of treatments for cyclical data," *ScholarWorks at California State University, Fullerton*, 2021. Online; accessed 2026.

[7] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988.

[8] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997.

[9] A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," in *AAAI Workshop on Learning for Text Categorization*, 1998.

[10] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *Proceedings of the 22nd ACM SIGKDD*, 2016.

[11] A. Y. Ng, "Feature selection, l1 vs. l2 regularization, and rotational invariance," in *ICML*, 2004.

[12] A. Genkin, D. Lewis, and D. Madigan, "Large-scale bayesian logistic regression for text categorization," *Technometrics*, 2007.

[13] L. Bottou, "Large-scale machine learning with stochastic gradient descent," *Proceedings of COMPSTAT*, 2010.

[14] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems*, vol. 24, 2011.

[15] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.

[16] P. Vuttipittayamongkol, C. de la Fuente, and F. Herrera, "On the class overlap problem in imbalanced data classification," *Knowledge-Based Systems*, vol. 212, p. 106638, 2021.

[17] J. Stefanowski, "Overlapping, rare examples and class decomposition in learning classifiers from imbalanced data," *Emerging Paradigms in Machine Learning*, pp. 277–306, 2013.

[18] A. Khurana and O. P. Verma, "Optimal feature selection for imbalanced text classification," in *Proceedings of the International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 1–6, IEEE, 2022.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *Proceedings of NAACL-HLT*, pp. 4171–4186, 2019.

[21] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, "Deep learning–based text classification: A comprehensive review," *ACM Computing Surveys*, vol. 54, no. 3, pp. 1–40, 2021.