Valerie Holtz
Matrikelnummer: IU14082683

10.04.2025

Project: NLP (DLBAIPNLP01)

PROF. DR. SIMON MARTIN

Task 3: Text Classification for Topic Modeling

Project Report

Table of Contents

Task 3: Text Classification for Topic Modeling
Project Report

1. Introduction

The goal of this project was to develop a robust text classification system for topic modelling, using the Reuters-21578 dataset. While the dataset is originally multi-label in nature, the project requirements specified a multiclass setup, where each document must be assigned exactly one label. To meet this constraint, I transformed the dataset by selecting the most frequent category per document as its sole label. In case of ties, one of the top labels was chosen randomly.

The project was iterative and included extensive experimentation with various model architectures, from traditional machine learning algorithms such as Naive Bayes and Random Forests to more advanced models like XGBoost and transformer-based architectures like DistilBERT. The main motivation was not only to classify the Reuters articles accurately but also to explore how well these models generalize to a completely different domain—social media, using a Twitter vaccine concern dataset.

Throughout the process, I faced challenges related to preprocessing SGML data, handling class imbalance, label inconsistencies, runtime errors during training, and adapting models across different text genres. These issues shaped the path and decisions made during the development and evaluation phases.

The project can be found here https://github.com/valerieholtz/topic_modeling/tree/main.

2. Dataset and Preprocessing

2.1 SGML Parsing and Data Extraction

The Reuters-21578 dataset was provided in SGML format, which is not directly supported by most modern NLP libraries. Issues such as invalid characters, encoding issues, and difficulty extracting labels were encountered. To extract text and label data, I found a custom parser on a GitHub project that handled encoding inconsistencies, invalid tags, and structural irregularities in the Reuters-21578 dataset. This parsing step was complex and consumed significant time due to the dataset's non-standard format.

Due to performance issues when training locally, I also migrated the model training to Google Colab, which offered better GPU/TPU support.

Text Preprocessing and Vectorisation

I implemented a preprocessing pipeline that included tokenization, lowercasing, stopword removal, and stemming. This cleaned and normalized text was then vectorized using TF-IDF transformation, converting each document into a numerical representation suitable for machine learning models.

Multi-Label to Multi-Class Conversion

Each article in the original dataset had multiple associated categories. Since the classification task required only one label per document, I selected the most frequent label for each document. When ties occurred, one label was randomly chosen. This process simplified the dataset but also introduced potential label noise and removed useful co-labeling information.

2.2 Dataset Characteristics

An important characteristic of the Reuters-21578 dataset is the high number of unique classes. After transformation, the dataset contained 442 unique classes. This extremely high number of categories made classification significantly more complex. Many classes were sparsely represented, with some having only one or two samples. Such imbalance was a central challenge for all models evaluated in this project.

Dataset Complexity

This level of granularity in the dataset significantly increases the classification challenge. Not only do models have to differentiate between many similar or overlapping topics, but the vast majority of these categories are highly imbalanced, with some having only a few samples. Not only the traditional models (e.g., Naive Bayes and Random Forest) struggled with this, even more sophisticated models like XGBoost or BERT had trouble generalising to the data.

As a result, the performance bottleneck is not just due to model choice but also stems from the underlying structure and distribution of the data.

Handling Class Imbalance

Class imbalance was evident in the Reuters dataset. After converting my dataset to a multiclass format, I inspected the class distribution. For example, the top 20 categories by frequency in the dataset are as follows. Note, that most classes had fewer than 10 samples:

```
Top 20 categories by frequency:
         Category  Count
1             usa   8141
5            earn   2072
4             acq   1241
12             uk    816
11         canada    613
30          japan    505
95       money-fx    300
61  west-germany    283
25          crude    249
70         france    238
68       interest    232
35          grain    167
56          trade    148
6          brazil    143
53      australia    138
27   switzerland    113
14           ship    109
10  money-supply    105
66          china     98
88   netherlands     92
```

To address this, I applied class weights and filtered extremely rare classes. SMOTE was also tested for traditional models in a first approach, but not applied to BERT due to its complexity. Overall, I suspect label imbalance remained the single greatest bottleneck to performance.

3. Model Development Baseline

3.1 Naive Bayes Baseline

Naive Bayes was selected as an initial baseline model due to its simplicity, interpretability, and efficiency—qualities that make it a common choice in many text classification tasks. However, while it performs well in balanced or moderately imbalanced scenarios, it is generally not a good fit for highly imbalanced datasets (Pedregosa et al., 2011). In my case, the significant class imbalance in the Reuters dataset caused Naive Bayes to heavily favor dominant categories, resulting in poor recall and F1-scores for underrepresented classes.

- Accuracy: 0.42
- Macro F1-score: 0.02
- Weighted F1-score: 0.34

These results show the difficulty of handling an imbalanced dataset using a simple model like Naive Bayes.

3.2 Random Forest (Untuned)

Given the limitations of Naive Bayes, I shifted to a Random Forest classifier with class_weight='balanced'. My initial (untuned) Random Forest model achieved:

- Accuracy: 0.43
- Macro F1-score: 0.06
- Weighted F1-score: 0.37

The model's performance on minority classes remained low, as expected given the skewed distribution.

3.3 Tuned Random Forest

To address the shortcomings of the initial Random Forest model, I conducted hyperparameter tuning using RandomizedSearchCV. The key hyperparameters tuned included (Pedregosa et al., 2011):

- n_estimators: Number of trees in the forest. A larger number of trees can reduce variance.
- max_depth: Maximum depth of the trees. Limiting depth helps prevent overfitting.
- min_samples_split: Minimum number of samples required to split a node. Higher values help avoid splits based on very few samples.
- min_samples_leaf: Minimum number of samples required at a leaf node. This prevents the creation of leaves with too few samples.
- max_features: Number of features to consider at each split. Options such as 'auto', 'sqrt', and 'log2' were explored.
- Bootstrap: Whether to use bootstrap samples when building trees.

After tuning, the best Random Forest model achieved improved metrics:

- Accuracy: 0.44
- Macro F1-score: 0.09
- Weighted F1-score: 0.44

Despite these improvements, the overall performance on minority classes remained suboptimal, as indicated by the relatively low macro averages.

3.4 XGBoost

After tuning and testing the Random Forest model, I attempted to improve classification performance by training an XGBoost model. XGBoost is a powerful gradient boosting framework known for its high accuracy, scalability, and ability to handle imbalanced datasets when tuned properly (Chen & Guestrin, 2016).

However, the dataset's extremely high number of classes and imbalanced distribution posed a challenge even for XGBoost. Additionally, technical hurdles arose during implementation—such as needing to filter out rare classes and ensure label consistency between training and test sets—before the model could be trained successfully.

Despite these preparations, the model's results showed only incremental improvements over Random Forest, particularly for the majority classes. The final evaluation metrics were

- Accuracy: 0.46
- Macro F1-score: 0.07
- Weighted F1-score: 0.43

The low macro average again confirmed that performance across all classes—especially minority ones—remained poor. Given these results and the growing complexity of the data, I decided to shift my focus toward transformer-based models such as BERT, which are better equipped to capture semantic nuances in text and have demonstrated state-of-the-art results in NLP tasks (Wolf et al., 2020).

4. Model Development Transformer

4.1 Transformer-Based Classification with BERT

Given the challenges faced by traditional models, my next step was to experiment with transformer-based models like BERT. I choose the lighter version, DistilBERT. Transformers have demonstrated strong performance on text classification tasks due to their ability to capture contextual nuances (Wolf et al., 2020). The task was to fine-tune a BERT model on the same multiclass version of the dataset. I tried to incorporate advanced hyperparameter tuning (e.g., learning rate scheduling, early stopping, gradient accumulation) to optimize performance and compare the BERT model's performance with the other models, particularly for minority classes.

Preprocessing Adjustments and Label Challenges

Before training BERT, I needed to ensure label consistency, since BERT expects a single integer label per instance ranging from 0 to *num_labels - 1*. My dataset originally had multi-labels, which were converted to multiclass by selecting the most frequent label per document. However, I encountered several issues:

Some labels appeared only once in the dataset, which posed a problem during stratified train-test splitting. I addressed this by removing rare labels (e.g., labels with fewer than 5 samples).After filtering, I re-encoded labels using LabelEncoder, but inconsistencies still appeared in label values, especially when the range was not continuous or started from a non-zero number. I verified and enforced that the minimum label was 0, the maximum was num_labels - 1, and that all labels were within this range. These steps were necessary to avoid runtime errors related to classification head shape mismatches in the model.

BERT Model Results

After extensive preprocessing, label range validation, and debugging, I successfully trained a DistilBERT-based classifier on my transformed multiclass Reuters dataset. The training process ran for 6 epochs with early stopping, and the model achieved the following performance:

- Accuracy: 0.5
- F1-score: 0.48
- Precision: 0.48
- Recall: 0.5
- Epochs: 6

This model outperformed previous approaches, especially on underrepresented classes, but still remain unsatisfying.

Saving, Loading, and Predicting with BERT

After successfully training the BERT model, I wanted to ensure that I could reuse the model without needing to retrain it each time. This is particularly useful for reproducibility and for scenarios where inference on new, unseen data is required.

Once the training process was completed, I saved the trained model and tokenizer to Google Drive. This allows persistent storage beyond the current Colab session. This saves all necessary model components, including the architecture configuration and fine-tuned weights. To avoid retraining, I can load the model and tokenizer from the saved path at any point.

Making Predictions on Unseen Text

To assess the generalization capability of the model, I used it to predict categories for a series of manually selected, clearly worded test texts (e.g., short news-style sentences). I defined a modular function to generate predictions using a text classification pipeline. This function handles tokenization, inference, and label decoding and let's us make prediction on unseen data. The model handled these examples reasonably well, mapping them to coherent categories from the Reuters taxonomy.

Evaluation on the Twitter Vaccine Concern Dataset

Next, I tested the model on a real-world, out-of-domain dataset: the Twitter Vaccine Concern Dataset, which consists of nearly 10,000 tweets labeled with vaccine-related concerns such as "side-effect," "political," and "pharma." These labels do not overlap semantically or structurally with the Reuters categories. When applying my Reuters-trained model directly to this dataset, performance was poor. Many predictions defaulted to frequent Reuters categories (e.g., "acq"), demonstrating that the model did not generalize well to informal or user-generated content.

4.2. Transfer Learning: Fine-Tuning BERT on Twitter Dataset

To strengthen the generalizability of my pretrained BERT model (initially trained on the Reuters newswire dataset) and improve performance, I decided to fine-tune it on a new dataset composed of vaccine-related tweets. This allowed to further assess how well the model performs on informal, short-form social media text in comparison to news articles.

The Twitter dataset contains user-generated content and uses a different tone, vocabulary, and structure than the Reuters dataset, making it a valuable benchmark for transfer learning.

Transfer Learning: Fine-Tuning on Twitter

Fne-tuned the Reuters-trained model on the Twitter dataset included the following preprocessing steps:

- Removing label classes with <2 samples
- Re-encoding labels
- Cleaning malformed rows
- Casting labels to int64

4.2.1 Challenges in Adapting the Model to Twitter Data

While fine-tuning my pretrained DistilBERT model on the Twitter vaccine concern dataset, I encountered a range of technical and practical challenges that required iterative debugging and careful handling.

1. Label Inconsistencies and Class Imbalance
The Twitter dataset originally contained 194 unique labels, many of which appeared only once. This caused issues when using stratified train-test splitting, resulting in errors due to insufficient representation of certain classes. I filtered out all classes that appeared only once and re-encoded the remaining labels. This ensured a cleaner label distribution and a contiguous numerical label range suitable for classification.

2. Model Loading and Classifier Shape Mismatch
Loading the previously saved DistilBERT model, which had been trained on the Reuters dataset, led to a shape mismatch in the classification layer. This was due to a difference in the number of classes between the Reuters dataset and the Twitter dataset. **I** replaced the model's classification head to match the number of classes in the Twitter dataset, allowing for compatibility with the new task.

3. Label Type Issues During Training
During training, I encountered runtime errors related to the data type of the labels. The loss function expected labels of a specific integer type, but the input data did not conform. **I** explicitly cast the label data to the appropriate format required by the loss function, which resolved the error and allowed training to proceed.

4. Data Import and Formatting Problems
The structure of the Twitter CSV file presented issues during import. Some rows were malformed due to inconsistent quotation marks and delimiters, leading to errors and incorrect parsing.**I** manually cleaned and reconstructed the dataset, ensuring that only valid rows with proper formatting and non-empty labels were included in the training data.

5. Incorrect Predictions for Empty Inputs
When testing the model, some predictions were generated for tweets that contained no text, typically defaulting to high-frequency categories. **I** ensured that all rows with empty or invalid text were filtered out before inference to prevent such spurious predictions.

After overcoming these hurdles, performance stabilized.

4.2.2 Training and Performance

The model was fine-tuned for 5 epochs with early stopping. Evaluation metrics:

- Accuracy: 0.563
- Precision: 0.452
- Recall: 0.563
- F1-score: 0.497

This showed that fine-tuning enabled the model to adapt to informal, short-form content.

These results show that the pretrained BERT model exhibits moderate performance when fine-tuned on a domain-specific dataset, but still not meeting expectations, indicating potential confusion across certain categories. This highlights the challenges of domain adaptation in NLP tasks, particularly when the target domain (tweets) differs significantly from the source domain (formal news articles).

5. Model Performance Summary

Over the course of the project, I trained and evaluated multiple models ranging from classical machine learning algorithms to modern transformer-based architectures. Below is a summary of each model's performance based on its accuracy, weighted average, macro average, and F1 scores.

1. Naive Bayes (Baseline)

- Accuracy: 0.42
- Macro Average F1-score: 0.02
- Weighted Average F1-score: 0.34

As expected, Naive Bayes struggled significantly due to the high class imbalance and simplistic assumptions about feature independence.

2. Random Forest (Untuned)

- Accuracy: 0.43
- Macro Average F1-score: 0.06
- Weighted Average F1-score: 0.37

Random Forest improved slightly on the baseline but still had low recall and poor macro performace, indicating difficulty with rare classes.

3. Tuned Random Forest (RandomizedSearchCV)

- Accuracy: 0.44
- Macro Average F1-score: 0.09
- Weighted Average F1-score: 0.44

Hyperparameter tuning led to minor gains in performance. However, the model continued to underperform on underrepresented classes.

4. XGBoost

- Accuracy: 0.46
- Macro Average F1-score: 0.07
- Weighted Average F1-score: 0.43

XGBoost slightly outperformed Random Forest, showing better generalization. Still, macro metrics remained low, reflecting ongoing struggles with imbalance and noise.

5. BERT (Trained on Reuters Dataset)

- Accuracy: 0.51
- Precision: 0.48
- F1-score: 0.48

The transformer model outperformed all previous approaches, but not by far. While the gains were not dramatic, BERT demonstrated improved ability to generalize over diverse classes.

6. BERT (Pretrained on Reuters, Fine-Tuned on Twitter Dataset)

- Accuracy: 0.56
- F1-score: 0.50
- Precision: 0.45
- Recall: 0.56

Fine-tuning BERT on a new domain (Twitter) produced the best overall performance, though still moderate. The model achieved an F1 score just under 0.50, revealing the difficulty of adapting to noisy, informal text.

None of the models performed exceptionally well, especially on macro-level metrics. Transformers outperformed classical models, but only modestly. The consistent pattern across models reveals that data-related issues due to imbalance, label noise and class fragmentation were likely the primary limitation rather than model architecture.

6. Summary of Key Challenges

Throughout the course of this project, I encountered several critical challenges, both on the data side and the modeling side. These issues shaped the direction of the work and significantly influenced model performance.

1. SGML File Format and Preprocessing Complexity

The Reuters-21578 dataset came in an SGML format, which is not natively supported by modern data libraries. Parsing this format proved to be one of the earliest hurdles in the project. I had to adapt a custom parser to extract the raw text and category tags, clean up inconsistent formatting, encoding issues, and nested elements and normalize and preprocess the extracted text through tokenization, lowercasing, stopword removal, and stemming.

This step was non-trivial and consumed considerable development time, highlighting how data ingestion alone can be a major bottleneck in NLP projects.

2. Highly Imbalanced Dataset with a Large Number of Classes

The dataset included 442 unique classes, many of which had very few examples. This created a significant class imbalance problem, affecting model learning capacity, as dominant classes overwhelmed minority ones. The stratification process in training/testing splits, often breaking due to classes with only one sample. I addressed this partially by removing rare classes and trying oversampling techniques, but these approaches were limited in scope due to time and complexity constraints.

3. Multi-Label to Multi-Class Conversion

Originally, the Reuters dataset was multi-label. Since the project required a multi class approach, I converted the dataset into a multiclass format using the most frequent label per sample. This may have been a crucial design flaw, because in cases of label ties, a random class was assigned. This introduced label noise and inconsistency, likely confusing models during training. Important co-labeling information was lost, which may explain why even transformer-based models failed to reach high performance.

4. Model Performance and Generalization

Across all model types the performance remained moderate at best. BERT, despite its strength, struggled with generalization, especially on informal text from Twitter. Even the fine-tuned BERT model, trained on the Twitter dataset, achieved only around 0.49 F1-score. This underwhelming result emphasises the complexity of the dataset and the potential flaws in label preparation.

5. Transformer Infrastructure and Pipeline Setup

Working with transformer models like BERT introduced several technical challenges that required careful handling throughout the project. First, the input text needed to be tokenized and padded to fixed lengths to meet the model's input format requirements. In addition, class labels had to be mapped to integer IDs and explicitly cast to the correct data type (specifically int64) to prevent runtime errors during training.

Training on GPUs brought its own set of complications. I frequently encountered CUDA memory and device-side errors, which often halted execution without clear tracebacks. To address this, I had to configure the environment using settings such as CUDA_LAUNCH_BLOCKING=1, apply manual random seeds for reproducibility, and reduce batch sizes to minimize GPU memory overload.

Moreover, when reusing a pretrained BERT model for domain adaptation, such as transferring from the Reuters dataset to the Twitter vaccine concern dataset, additional adjustments were necessary. In particular, I had to modify the model’s output layer to match the number of classes in the new task. This process of architectural alignment was critical to ensure compatibility and avoid shape mismatch errors during fine-tuning.

6. Hyperparameter Tuning and Experimentation Gaps

Choosing the appropriate hyperparameters for models such as Random Forest, XGBoost, and BERT presented another significant challenge during the project. Time and resource limitations meant that I was only able to conduct basic tuning in most cases. For instance, RandomizedSearchCV was used for the Random Forest model to explore a limited subset of possible hyperparameter combinations. However, for the BERT model, I was not able to implement more advanced tuning strategies, such as grid search, learning rate scheduling, warmup steps, or gradient clipping, which are often critical for achieving optimal performance in deep learning settings.

As a result, the absence of more exhaustive fine-tuning likely constrained the models’ ability to generalize, particularly in complex classification scenarios involving imbalanced and high-dimensional label spaces. This limitation is an important consideration when interpreting the results and serves as a key area for improvement in future iterations of the project.

7. Class Rebalancing and Under-Explored Strategies

While I implemented some foundational techniques to address class imbalance such as class weighting and filtering out rare classes for traditional models, there remained significant opportunities to enhance this aspect of the pipeline. More advanced strategies like SMOTE could have been explored to improve model performance, particularly on underrepresented classes.

8. Retraining Infrastructure

Reusing a pretrained model and adapting it for a new dataset was a multi-step process that presented several technical hurdles. First, I had to align the label structures between the source and target datasets, ensuring that the classification head of the model matched the number of classes in the new task. This involved replacing or modifying the model's output layer to reflect the updated label space.

Next, the Twitter dataset had to be transformed into a format compatible with the model pipeline. This included careful preprocessing and re-encoding of labels to fit the expected input structure. Throughout this process, I also had to address potential issues such as mismatched tensor shapes and incorrect label encodings, both of which could easily lead to runtime errors during training. Overall, adapting the model for cross-domain transfer required extensive debugging and careful orchestration of every component in the training pipeline.

7. Conclusion and Future Directions

Despite extensive experimentation with four different model architectures - Naive Bayes, Random Forest, XGBoost, and two DistilBERT models - none of the models produced highly satisfactory results. Even BERT, often considered state-of-the-art for text classification, performed only moderately and struggled to generalize to other text types such as informal Twitter data.

The core challenge appears to lie with the dataset itself. The Reuters dataset has 442 unique classes, many of which are highly imbalanced, underrepresented, or overlapping in meaning. My decision to convert the original multi-label format to a multiclass format likely caused further issues. While this simplification was necessary for initial model training, it may have introduced noise or arbitrary label assignments, ultimately confusing the models.

In adapting the models to the Twitter vaccine dataset, I encountered further obstacles. These included data formatting issues, label noise, and the inherent informality and ambiguity of tweets. My pretrained BERT model, even after fine-tuning, showed only modest performance when evaluated on this new domain.

Despite the results, the project offered a learning experience, especially in the things that went wrong:

- Preprocessing: I implemented robust pipelines for both structured datasets (Reuters) and noisy, user-generated content (Twitter).
- Modeling: I applied traditional ML classifiers and transformer-based models, with full training, evaluation, and hyperparameter tuning.
- Handling Class Imbalance: I tried filtering rare classes. However, more advanced strategies remain to be explored.
- Debugging and Fine-tuning: Iresolved multiple issues including CUDA memory errors, label mismatches, and optimizer configuration for transfer learning.
- Model Management: I successfully saved and reloaded models for later inference and built pipelines to predict labels on unseen examples.

The best-performing model, a BERT variant fine-tuned on Twitter data, achieved:

Accuracy: 0.563
Precision: 0.452
Recall: 0.563
F1-score: 0.497

While not a strong result, it suggests that there is potential, especially with better data and deeper fine-tuning. The commonly cited saying "garbage in, garbage out" seems to apply here. Even powerful models cannot overcome poorly structured or misleading training labels.

Lessons Learned and Future Directions

To enhance the performance of future iterations of this project, several promising directions can be explored. One of the most crucial recommendations is to rethink the labeling strategy. Flattening a multi-label dataset into a multiclass format, while sometimes convenient, can lead to a loss of information and result in misleading input for the model. A more effective approach would be to retain the original multi-label structure and develop models specifically designed to handle multiple categories per document, thereby preserving the full nuance of the data.

Another significant improvement lies in more strategic handling of imbalanced data. The extreme skew in class distribution likely served as a major bottleneck to model performance. To mitigate this, future work could incorporate advanced techniques.

Active learning presents another powerful strategy. By allowing the model to identify and request labels for the most uncertain or informative samples from large pools of unlabeled data (such as Twitter), active learning can improve model efficiency and overall generalization in a resource-conscious manner (Schröder, Müller, Niekler, & Potthast, 2021).

Moreover, more intensive fine-tuning practices could significantly improve results. In future iterations, exploring more expressive transformer variants might also yield better results, especially when paired with appropriate regularization.

8. Sources

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830. https://scikit-learn.org/stable/

Schröder, C., Müller, L., Niekler, A., & Potthast, M. (2021). *Small-Text: Active learning for text classification in Python*. https://doi.org/10.48550/arXiv.2107.10314

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., ... & Rush, A. M. (2020). Transformers: State-of-the-art natural language processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45. https://doi.org/10.18653/v1/2020.emnlp-demos.6