Text Mining Project

Group 29 - Sanne Breed, Ioana-Teodora Gomoi, Gintarė Rutkutė, Hannah Sallai
Repository Link: https://github.com/HannahUsesTheGitHub/TM/tree/main

01 Introduction

The world has become a place of overwhelming amounts of information that can drive decision-making across multiple industries. However, extracting meaningful insights from raw text requires advanced natural language processing (NLP) techniques that imply more than just simple keyword matching. In this context, this project explores the integration of three key NLP tasks: named entity recognition and classification (NERC), sentiment analysis and topic analysis to gain a deeper understanding of textual data.

02 Named Entity Recognition and Classification (NERC)

→ Pre-processing and Feature Selection
For the Named Entity Recognition and Classification (NERC) we used a dataset from Kaggle [9].
The pre-processing steps were kept simple, with splitting the sentences into separate words. Within the feature extraction the words themselves were transformed into lower case, however, the information weather a word was initially capitalized was preserved as this might give important insights for recognizing entities.
Standard features for this task were used, namely the word in lower case, the word suffix of three and 2 characters, whether the word is capitalized, is in title script, or is a digit, the Part of Speech (PoS) tag of the word and the first 2 characters of the PoS tag. For the PoS tagging, the .tag_ feature from spacy was used. Additionally, to the features of the current word, the lowercase form, if the word is capitalized or in title case, the PoS tag and the first two characters of the PoS tag were added for the word preceding and succeeding the current word. If no one preceded the current one, the current one, the BOS label was added to the features instead to signal the beginning of the sentence. Similarly, if there was no following word in a sentence, the EOS (end of sentence) tag was added to the features.
A test set (20% of the data) was split from the data, the remaining data was split into a validation set (20% of the remaining data) and a training set. The test set was only used to measure the performance of the final model, the validation set was used for hyperparameter tuning and selecting the best model. In the end the best model was also evaluated on the test data that was provided for the assignment.

->Data
The NERC test data consisted of only 12 sentences, which corresponds to 164 labeled instances. The majority class is O with 71.34% of the dataset, followed by I-per with 7.32%, then B-per with 6.1%, B-org with 4.27%, I-art with 3.66% and I-org with 3.05%. The rest of the classes all make out below 2% of the dataset each. There are 9 classes in total. The test set included 9,578 sentences which yielded 209,994 tokens/labels, the biggest class of the labels was O with around 84.74%. The next 5 classin in descending order were B-geo with 3.59%, B-tim with 1.95%, B-org with 1.91%, I-org with 1.53% and I-per with 1.64%. In terms of size, the validation set included 7,662 sentences, corresponding to 166,841 word tokens. Again, the majority class was O with 84.62%, succeeded by B-geo with 3.53%, B-tim with 1.97%, B-org with 1.95%, B-per with 1.61% and B-gpe with 1.6%. As the biggest set, the training data includes 30,650 sentences, which translates to 670,156 BIO labels. Unsurprisingly O is the biggest class with 84.62% of the tokens. Next are B-geo including 3.62% of the labels, B-tim with 1.93%, B-org with 1.92%, I-per with 1.66% and B-per with 1.64%. All sets are unbalanced with O as the majority class and B-geo including a bigger sample than the other B and I labels. We decided against artificially balancing the classes. While this might result in better results for the dataset that is trained on, this artificial balance would not correspond to any real world scenario where the majority of words would indeed not be named entities (corresponding therefore to an O in terms of BIO tags). The distributions within the training, test and validation sets are all similar, making accurate performance metrics possible.

→ Method and Model
For the actual NERC task, we trained a conditional random field (CRF) model with the modules from sklearn. Conditionally random field for IOB sequences maps sequence dependencies between tokens. Therefore, labels are not just assigned based on the current token, but based on sequential paths.
There was extensive hyperparameter tuning for the CRF model using a grid search method. While the feature engineering stayed constant, the search space fitted models with the optimization algorithms lbfgs (limited memory stochastic gradient descent), l2sgd (stochastic gradient descent with L2 regularization), ap (Averaged Perceptron), pa (Passive Aggressive). For both the L1 and L2 regularization parameters the values 0.2, 0.1, and 0.01 were tested. The models were trained with and without all_possible_transitions activated. If it is activated, transition features are generated that do not occur in the training set. All possible combinations of the named values were computed and compared by the weighted average of their f1-score (calculated for the validation set). The impossible combinations were skipped (for example l2sgd does not support L1 regularization). One hyperparameter that was not tunes is the maximum number of iterations, which was fixed and set to 100. Since the weighted average f1 score can be misleading due to the high class imbalance, we also optimized with regards to the macro average f1 score.
One limitation of the method is that more parameters were tested for the algorithms working with gradient descent (lbfgs and l2sgd), giving them more possibilities to achieve good results.

→ Analysis
The best model was achieved with the lbfg algorithm, an L1 regularization parameter of 0.2 and an L2 regularization parameter of 0.1. all_possible_transitions was set to the Boolean value True, so the feature was activated. For both the macro and the weighted average f1 score this was the highest performing setup.
This best model was evaluated first on the validation set and then on the test set. The values for the validation and the test set show a lot of similarity with some values even being slightly better for the test set. This is an indication that the model did not overfit to the training set. The following values are all from the evaluation on the test set (Fig.2.1). With 0.97, the model achieved a high overall accuracy. The macro average for the precision is 0.73, for the recall is 0.62 and for the f1-score is 0.66. With regards to the weighted average, the precision, recall and f1-score values are all 0.97. O reached the overall the highest precision, recall and f1-scores with 0.99 each, which was to be expected seeing as O is the majority class. Outliers with regards to especially low performance metrics were the classes I-art (precision: 0.29, recall: 0.1, f1-score: 0.15) as well as B-art (precision: 0.24, recall: 0.07, f1-score: 0.11). This is most likely doe to the low support of the art class in general, especially with regards to I-art, which only has 210 training instances in total and only 59 instances in the test set. For the same reason the B-nat and especially I-nat classes do not perform well, with B-nat having a precision of 0.53, a recall of 0.40 and an f1-score of 0.47 with a support of 72 in the test and 135 in the training set. I-nat performs even worse with a precision of 1.0, a recall of 0.4 and an f1-score of 0.57. It has a support of only 24 in the training data and 5 in the test data. While the precision score of 1 looks perfect at a first glance, this probably only means that none of the 5 instances were false positives. The metrics for this minority class are most likely not better than random guessing would be.

When visualizing the weight for the CRF model (Fig.1) it can be seen that the model learned high positive correlations from B-eve to I-eve, from B-geo to I-geo, B-grp to I-grp, from B-art to I-art, from B-nat to I-nat, from B.org to I-org, from B-per to I-per and from B-tim to I-tim, which are aspects which we wanted the model to realize. Furthermore, it can be seen that especially negative connections are formed from O to I-geo, I-org, I-per and I-time, which makes sense since O would be followed by a B, which only then would be followed by an I label. Moreover, B-grp is not often followed by B-grp, indicating that groups often have a one-word name, from B-grp to I-org, signaling that group entities are not often followed directly by organization entities. The negative weights from B-per and I-per to B-per might indicate that in the training data, persons were often only having a single word name (as supposed to a full name consisting of first and last name directly following each other).

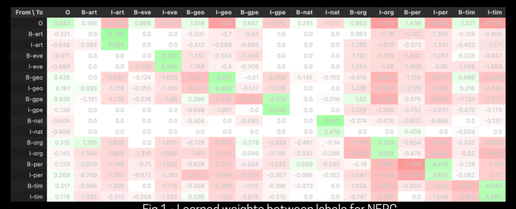

Fig.1 - Learned weights between labels for NERC

Lastly, the model was tested on the test data that was specifically provided for NERC (Fig.2.2). Here the model had a lower accuracy of 0.84. The weighted averages are 0.82 for the precision, 0.84 for the recall and again 0.82 for the f1-score. These values are probably highly skewed due to the metrics of the majority class O, which has a precision of 0.91, a recall of 0.96 and an f1-score of 0.93. Macro averages show a more balanced view on the data and are worse that the weighted averages with precision and f1-score only lying at 0.43 and recall being slightly better with 0.45. However, this data might be impacted by the three classes that do appear in the test set and therefore return 0 for precision, recall and f1-score. Taking this into account and ignoring the values where there is a zero-division, the weighted averages are precision: 0.87, recall: 0.84. The macro averages improve even more with precision: 0.52, recall: 0.6. The f1 score does not change, since these empty classes are still (wrongly) predicted by the system, even though they don't exist in the test set. An outstandingly good performance can be seen got I-geo, where precision, recall and f1-scores are all 1, indication perfect performance. However, since the support for this class in only 2, this could also be based on luck. Low performance can especially be observed in classes that already performed badly on the prior test set, namely B-art and I-art. All of the values for art are 0, indicating that no instance was classified correctly. The second biggest class in the test set, I-per performed relatively well with a precision of 1, a recall of 0.58 and an f1-score of 0.74. The equivalent B-per has well and more balanced results with a precision of 0.73, a recall of 0.8 and a f1-score of 0.76. In general, the small size of the test set makes the performance evaluation less significant.

Report on test set:

	precision	recall	f1-score	support
B-art	0.24	0.07	0.11	72
B-eve	0.58	0.40	0.47	55
B-geo	0.85	0.90	0.87	7530
B-gpe	0.96	0.93	0.94	3198
B-nat	0.53	0.33	0.41	27
B-org	0.78	0.70	0.73	4018
B-per	0.83	0.81	0.82	3292
B-tim	0.93	0.89	0.91	4087
I-art	0.29	0.10	0.15	59
I-eve	0.34	0.25	0.29	40
I-geo	0.83	0.78	0.80	1488
I-gpe	0.81	0.57	0.67	23
I-nat	1.00	0.40	0.57	5
I-org	0.80	0.78	0.79	3416
I-per	0.84	0.89	0.87	3390
I-tim	0.83	0.78	0.81	1355
O	0.99	0.99	0.99	177939
accuracy			0.97	209994
macro avg	0.73	0.62	0.66	209994
weighted avg	0.97	0.97	0.97	209994

Fig.2.1 - Performance metrics on extended, personally-developed test set

	precision	recall	f1-score	support
B-art	nan	0.00	0.00	3
B-geo	0.67	1.00	0.80	2
B-gpe	0.00	nan	0.00	0
B-org	0.43	0.43	0.43	7
B-per	0.73	0.80	0.76	10
B-tim	0.00	nan	0.00	0
I-art	nan	0.00	0.00	6
I-geo	1.00	1.00	1.00	2
I-org	0.43	0.60	0.50	5
I-per	1.00	0.58	0.74	12
I-tim	0.00	nan	0.00	0
O	0.91	0.96	0.93	117
accuracy			0.84	164
macro avg	0.52	0.60	0.43	164
weighted avg	0.87	0.84	0.82	164

Fig.2.2 - Performance metrics on provided test set

03 Data Pre-processing

To create a balanced training dataset for sentiment classification, we combined three datasets from Kaggle, each representing a different topic: books, movies, and sports. For books, we used the "Amazon Kindle Book Review for Sentiment Analysis"[1] dataset, for movies, the "IMDb Movie Reviews Genres Description and Emotions"[2] dataset, and for sports, the "sentiment-analisis-sports"[3] dataset. All the preprocessing work was done in Apple Numbers using spreadsheet functions and manual filtering.
The original sizes of the datasets varied significantly: the book dataset contained 12,000 entries, the movie dataset had 19,315 entries, and the sports dataset was much larger, with 72,515 entries.
We began by cleaning each dataset individually. This involved removing any duplicate entries and deleting excessive columns such as metadata, review summaries, or details about specific movies or matches that were not relevant to sentiment classification.
Once the data was cleaned, we focused on standardising the sentiment labels. The sports dataset already contained sentiment annotations, but these included the recipient of the sentiment and, in some cases, multiple sentiments per review. To simplify this, we removed recipient tags and excluded any entries with more than one sentiment, retaining only those with a single, clearly defined sentiment. The movie and book datasets did not include sentiment labels, so we derived them from user-provided star ratings. In the movie dataset, ratings ranged from 1 to 10. We mapped these to sentiment classes as follows: 1 to 3 was considered negative, 4 to 6 neutral, and 7 to 10 positive. In the book dataset, ratings were on a scale from 1 to 5, with 1 to 2 labeled as negative, 3 as neutral, and 4 to 5 as positive. These mappings were implemented using formulas directly in Apple Numbers. After assigning sentiment labels, we added a new column to each dataset to indicate the topic of each entry, either "book", "movie", or "sports".
After labelling and standardising the datasets, we needed to combine them into a single, balanced dataset. Since the book dataset had the fewest usable entries (12,000), we used it as the reference point and matched the other datasets to this size. Our goal was to ensure an even distribution across both topics and sentiment classes. To achieve this, we manually selected 4,000 negative, 2,000 neutral, and 6,000 positive entries from the book dataset, 4,000 negative, 5,000 neutral, and 3,000 positive entries from the movie dataset, and 4,000 negative, 5,000 neutral, and 3,000 positive entries from the sports dataset. This resulted in a final dataset of 36,000 entries, with 12,000 reviews per topic and 12,000 per sentiment class.
Finally, we added a column that assigned a unique ID to each entry, starting from 0. This was done using a formula based on the row number. Once completed, the dataset was exported as a TSV file, ready to be used for training sentiment classification models.

04 Sentiment Analysis

Sentiment analysis is an important task in NLP that involves determining the emotional tone behind a text. This subcategory of text mining can be used for many applications, such as reputation management, where the general sentiment behind a brand or a newly implemented marketing strategy is analysed, helping to determine the public response to the changes and to offer insights into what the next steps of implementation would be, based on a positive or negative response.

→ Naive Bayes
For our project we decided to implement the machine learning-based Naive Bayes classifier [6] for sentence-level sentiment analysis due to its simplicity, efficiency and proven effectiveness in classification tasks. Naive Bayes is based on the Bayes' Theorem that is used to calculate the probability of a sentiment label (positive, neutral or negative) given the set of words in the sentence. Despite the "naive" assumption of feature independence, this model performs well in text classification. One of the reasons it does well is represented by the fact that the model leverages word frequencies to estimate sentiment probabilities and that the probabilistic output allows for confidence scoring, which is useful in uncertain cases.

Another reason behind our choices is that the model has high interpretability, meaning that it is easier to analyse why certain predictions are made, making the process of debugging and refinement smoother.
Furthermore, sentiment analysis involves processing large datasets, meaning that computational efficiency is crucial. In our project, the training data we used includes 36.000 instances, as we tried to reproduce a real life scenario of implementation. This is a pretty standard number of instances, therefore, the problem of computational efficiency is apparent. Keeping this in mind, the choice of doing sentiment analysis using a Naive Bayes model is further justified by the fact that it requires a single pass through the training data in order to compute the word probabilities, making it faster to train compared to more complex models such as neural networks. Its simplicity represents an advantage in our case, as it performs well even with limited computational resources, which is beneficial for research projects with constraints. Additionally, due to its simple architecture, it has a lower risk of overfitting on the training data provided, making it a generalizable model to use above multiple datasets.

→ Fine Tuning
In order to find which values of initialization of the model are best suited for our project, we implemented a grid search algorithm tasked with hyperparameter fine-tuning. The parameters we decided to look at are the following: minimum document frequency, which is indicative of the number of times a specific token will be taken into account (with values of two, five or ten), range of n-grams, which controls the range of contiguous sequences of n words included in the feature set (with values of (1,1), (1,2) or (2,2)), whether to use Bag of Words representation or TF-IDF representation (with boolean values of either True or False) and alpha, representing the smoothing factor of the Naive Bayes model (with values of 0.001, 0.01, 0.1 or 1.0). In order to evaluate the models with the sets of hyperparameters, 5-fold crossover was implemented, meaning that we divided our training data into five parts, using one part as validation data. Iteratively, each model with a different set of values is trained and tested on the five different versions of the training and validation datasets and then the macro F1-score and the weighted F1-score are used as evaluation functions. Based on these values, the best performing model was picked with the following set of parameters: minimum document frequency of 2, n-gram range of (1, 2), use of TF-IDF with value True and smoothing factor of 0.1.

→ Analysis
The performance of the grid search found model (Fig.3) is not exceptionally good, as it reaches an accuracy of 50%, meaning that it can only identify the right sentiment in half of the cases given to it. As a comparison of performance we picked another set of parameters to test when initializing the Naive Bayes model (Fig.4). The choice of parameters was based on previous experience we had gathered with the architecture, for the same task of sentiment analysis, when completing one of the assignments of the Text Mining Course, offered by Vrije Universiteit [4]. The chosen set of values (minimum document frequency of 5, n-gram range set on its default value, with use of TF-IDF of False, meaning that the representation of words and tokens is Bag of Words and smoothing factor left on its default value) has been analysed to be the best one performing the task of sentiment analysis given a refined dataset of airline tweets. For this reason, we decided to use it as a control measure, to check whether the grid search algorithm will find a better performing solution. The outcome of this comparison was unexpected, as the second set of values found for initializing the Naive Bayes model (Fig.4) ended up outperforming the best solution found by hyperparameter tuning (Fig.3) by 16%, meaning that it reached an accuracy of 66.7%. By comparing these two different behaviours, an observed insight is that our training data is not as generalizable or robust as we would like it to be, meaning that even if we have an equal distribution of domain-related text included inside of it, it is still not enough to make it representative for a real life scenario of implementation.
Furthermore, the difference in behaviour between the two sets of values that Naive Bayes was initialized with can be attributed to the fact that the first configuration (Fig.3) allowed for very rare terms (appearing in only two documents as defined by the minimum document frequency) to be included as features. These terms can be noisy, domain-specific or outliers, which ends up hurting generalization. As opposed to this method, the second configuration (Fig.4) filters out the rare terms by keeping those that appear in at least five documents, reducing noise and focusing on more stable, frequent terms that are likely more meaningful for sentiment.
A second difference can be represented by the set n-gram range. For the first configuration (Fig.3) the range (1,2) includes unigrams and bigrams, meaning that while bigrams capture phrases like "not good" or "very good", which are useful for sentiment, they might also introduce many low-level or irrelevant combinations, such as "the movie" or "the book", increasing dimensionality and noise. As compared to the first model, the second model (Fig.4) uses the default range (1,1), making it use only unigrams. This approach may miss some meaningful phrases, but unigrams often prove to be sufficient for sentiment analysis, also helping the configuration avoid noise and simplifying the feature space, reducing overfitting.
Another very important difference is represented by the method of representation used between the two models. On one hand, the first configuration (Fig.3) uses TF-IDF to represent its words, which downweights frequent terms, such as "the", and emphasizes rare but discriminative terms. This method isn't ideal for sentiment analysis, as common words, such as "love" or "hate", are highly informative and TF-IDF's downweighting can dilute their importance. On the other hand, the second configuration (Fig.4) uses Bag of Words representation, which preserves the raw term frequencies, meaning that it can be more effective for sentiment tasks, as just the count of sentiment words is directly correlated with the sentiment strength.
Lastly, the smoothing factor can have a big impact on the performance of the two models, as the first one (Fig.3) uses a smaller alpha of 0.1, that provides less smoothing, making the model more sensitive to the training, while the second one (Fig.4) uses the default alpha value of 1.0, which represents the default Laplace smoothing, being more conservative and helping handle unseen words better. Using Laplace smoothing improves generalization by avoiding extreme probabilities for rare terms.

== Best Weighted F1 Model ===
Settings: {'min_df': 2, 'ngram_range': (1, 2), 'use_idf': True, 'alpha': 0.1}
Weighted F1: 0.6598

	precision	recall	f1-score	support
negative	0.500	0.667	0.571	6
neutral	0.400	0.333	0.364	6
positive	0.600	0.500	0.545	6
accuracy			0.500	18
macro avg	0.500	0.500	0.494	18
weighted avg	0.500	0.500	0.494	18

Fig.3 - First Configuration Performance Metrics (Grid Search Solution)

Training with: min_df=5, ngram_range=(1, 1), use_idf=False, alpha=0.100
Macro F1: 0.6483, Weighted F1: 0.6483

	precision	recall	f1-score	support
negative	0.714	0.833	0.769	6
neutral	0.714	0.833	0.769	6
positive	0.500	0.333	0.400	6
accuracy			0.667	18
macro avg	0.643	0.667	0.646	18
weighted avg	0.643	0.667	0.646	18

Fig.4 - Second Configuration Performance Metrics

05 Topic Analysis

Topic classification is the task of assigning predefined categories or topics to pieces of text. It is widely used in applications such as content tagging, information retrieval, and organizing large volumes of text data. By automatically identifying the subject matter of texts, topic classification helps structure unlabelled data and supports downstream tasks like recommendation systems or targeted summarization.

→ Support Vector Machines
As our first model, we implemented an SVM classifier with a linear kernel and bag-of-words representation. SVMs are known for their robustness in high-dimensional spaces like text, where each word becomes a feature. The linear decision boundary is often sufficient for separating topics when the vocabulary carries strong signals. SVMs also tend to perform well on small to medium-sized datasets due to their ability to maximize the margins between classes, which improves generalization [7].
In this experiment, the SVM model, fine-tuned on the C parameter using grid search, achieved 77% accuracy on the test set, as seen in fig. 5. It was notably effective in correctly identifying sports-related content, reaching perfect recall in this class. However, it underperformed in detecting movie-related text, with a recall of only 50%. One reason for these errors could be the model's sensitivity to lexical overlap between classes. For instance, the sentence "It's surprising how a promising start could unravel into such a disappointing finish" was misclassified as sports, perhaps due to overlapping terminology like "a promising start" or the word "finish", which are likely to appear in sports commentary.

→ Multinomial Naive Bayes
For our second model, we trained a Naive Bayes classifier using bag-of-words representation. As discussed previously, Naive Bayes classifiers are useful due to their simplicity and surprising effectiveness in classification tasks [6].
In our experiment, the Multinomial Naive Bayes model was fine-tuned on the alpha hyperparameter using grid search, after which it achieved an accuracy of 83% on the test set as seen in fig. 6. It performed particularly well on the sports category, with perfect precision and recall, indicating that the class-specific vocabulary was distinctive and consistently learned. However, it was less reliable when distinguishing between movie and book, misclassifying more ambiguous examples. For instance, "The story had its moments, though some parts felt like they dragged on a bit." was classified as movie despite being book, likely due to similar language used in both types of reviews, using words like "story" and "dragged on". While not as nuanced as a contextual model, Naive Bayes serves as a strong baseline and highlights some of the limits of simple frequency-based approaches when dealing with overlapping topic boundaries.

→ RoBERTa
As part of our topic analysis task, we selected RoBERTa from the SimpleTransformers library as one of our main models. Although both BERT and RoBERTa performed well during experimentation in Lab 6, we opted for RoBERTa due to its improved architecture. RoBERTa builds upon BERT's foundation but benefits from extended pretraining on more data and removal of the Next Sentence Prediction (NSP) objective, which together make it more effective for text classification tasks, including topic classification [8].
RoBERTa uses Masked Language Modelling (MLM) to learn contextual embeddings by predicting masked tokens within input sequences. This focus on deep bidirectional context, without the need for sentence ordering, makes RoBERTa particularly suitable for sentence-level classification problems like ours.
For fine-tuning, we used the base configuration from Lab 6 [5], with a small adjustment: the evaluation frequency parameter (evaluate_during_training_steps = 32) was removed. We observed that keeping it led to premature stopping, likely due to early evaluation plateaus. Removing it allowed training to continue until the end of epoch 6.
On the development set, the model achieved an evaluation loss of approximately 0.0351 and a Matthews Correlation Coefficient (MCC) of approximately 0.9925. These scores suggest that the model learned highly discriminative features for our multi-class classification task.
On the test set, performance was similarly strong. As seen in the table(Fig.7), the model achieved an overall accuracy of 0.94, with perfect precision and recall for the "book" category, perfect precision for "movie," and perfect recall for "sports."
The model only misclassified one instance: sentence ID 16, "It's surprising how a promising start could unravel into such a disappointing finish.". This was predicted as sports but was labeled as movie. We believe this confusion is reasonable, as the sentence is ambiguous even to human readers and could refer to a sports event or a film without additional context.

→ Comparison
Each of the three models showed distinct advantages in the topic classification task, reflecting the trade-offs between model complexity, feature representation, and interpretability.
The traditional models, SVM and NB, both relied on a bag-of-words representation, yet approached the classification task differently. Naive Bayes, though simple, proved to be an effective baseline, especially for clearly delineated topics like sports. Its performance, however, suffered when topic boundaries became more ambiguous, as it lacks the ability to consider context beyond word frequency. SVM, with its capacity to draw more flexible decision boundaries in high-dimensional space, performed slightly worse overall but captured more subtle patterns in some cases. Still, both models struggled with examples where lexical overlap between topics introduced ambiguity.
RoBERTa, in contrast, demonstrated a significant leap in performance. By using contextual embeddings, it was able to account for semantic nuance and syntactic variation that the simpler models could not capture. It handled nearly all cases correctly and showed resilience in the face of ambiguous phrasing, outperforming the others across all metrics. Interestingly, the only error it made was on a sentence that was misclassified by all three models, highlighting how ambiguity in language can pose challenges even for advanced models.
Overall, these comparisons underline the limitations of frequency-based models in handling nuanced, short-text classification tasks and the improvements that come with using pre-trained transformers when data and computational resources allow.

	precision	recall	f1-score	support
book	0.71	0.83	0.77	6
movie	0.80	0.67	0.73	6
sports	1.00	1.00	1.00	6
accuracy			0.83	18
macro avg	0.84	0.83	0.83	18
weighted avg	0.84	0.83	0.83	18

Fig.6 - Naive Bayes Performance Metrics

	precision	recall	f1-score	support
book	1.00	1.00	1.00	6
movie	1.00	0.83	0.91	6
sports	0.86	1.00	0.92	6
accuracy			0.94	18
macro avg	0.95	0.94	0.94	18
weighted avg	0.95	0.94	0.94	18

Fig.7 - RoBERTa Performance Metrics

	precision	recall	f1-score	support
book	1.00	0.83	0.91	6
movie	0.75	0.50	0.60	6
sports	0.67	1.00	0.80	6
accuracy			0.78	18
macro avg	0.81	0.78	0.77	18
weighted avg	0.81	0.78	0.77	18

Fig.5 - SVM Performance Metrics

06 Conclusion and Limitations

In conclusion, this project successfully implemented an NLP pipeline, integrating Named Entity Recognition and Classification, Sentiment Analysis and Topic Analysis to extract structured insights from unstructured text. By combining these techniques, we explored how entities, sentiment and topics can be analysed to provide a more comprehensive understanding of textual data, which can be further useful for applications such as brand monitoring, reputation management and media trend detection.
Limitations of our project are represented firstly by the modest performance achieved by the Naive Bayes sentiment classifier (66.7% accuracy, 64.6% weighted average). To improve performance in the task of sentiment analysis, other Machine Learning methods such as transformer-based ones could be researched. Additionally, further pre-processing of data or exploration of more complete datasets could improve performance on the tasks, making the training set more representative for a real life scenario of implementation.

07 Division of work

Sanne Breed (2774417):
1) coding: SVM and Naive Bayes classifiers for topic classification (training, tuning, and analysing the models), also found the necessary datasets for topic analysis, 2) analysis: analysis of SVM and Naive Bayes, comparison between all topic analysis approaches, 3) poster preparation: Topic Analysis, help with formatting
Ioana-Teodora Gomoi (2800474):
1) coding: the entire Sentiment Analysis part (data formatting and cleaning, grid search method for hyperparameter tuning, Naive Bayes model) 2) analysis: analysis of results of Naive Bayes model and of Grid Search, 3) poster preparation: Sentiment Analysis, Introduction and Conclusion, formatting of the layout
Gintarė Rutkutė (2796845):
1) coding: data preprocessing and topic prediction with RoBERTa 2) analysis: analysis of RoBERTa model 3) poster preparation: data preprocessing and RoBERTa model, topic tables and some formatting
Hannah Sallai (2815516):
1) coding: the entire NERC part (data pre-processing, hyperparameter tuning, CRF model), 2) analysis of results of CRF on personal dataset and given dataset, analysis of fine tuning, 3) poster preparation: the whole section NERC

Sources

[1] https://www.kaggle.com/datasets/meetnagadia/amazon-kindle-book-review-for-sentiment-analysis?select=preprocessed_kindle_review+.csv
[2] https://www.kaggle.com/datasets/fahadrehman07/movie-reviews-and-emotion-dataset
[3] https://www.kaggle.com/datasets/carlosramosn/sentiment-analisis-sports
[4] Lab 3, Text Mining, Vrije Universiteit
[5] Lab 6, Text Mining, Vrije Universiteit
[6] Kibriya,A.M.,Frank,E.,Pfahringer,B.,Holmes,G.(2004).Multinomial Naive Bayes for Text Categorization Revisited. In: Webb, G.I., Yu, X. (eds) AI 2004: Advances in Artificial Intelligence. AI 2004. Lecture Notes in Computer Science(), vol 3339. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-30549-1_43
[7] X. Luo, "Efficient English text classification using selected Machine Learning Techniques," Alexandria Engineering Journal, vol. 60, no. 3, pp. 3401–3409, 2021, https://doi.org/10.1016/j.aej.2021.02.009.
[8] Y. Guo, X. Dong, M. A. Al-Garadi, A. Sarker, C. Paris, and D. M. Aliod, "Benchmarking of Transformer-Based Pre-Trained Models on Social Media Text Classification Datasets," in Proc. 18th Annu. Workshop Australasian Lang. Technol. Assoc. (ALTA), Virtual Workshop, Dec. 2020, pp. 86–91. [Online]. Available: https://aclanthology.org/2020.alta-1.10/
[9] https://www.kaggle.com/datasets/rajnathpatel/ner-data