Machine Learning for NE Recognition and Classification Report

Csenge Szabó

Abstract

This study investigates the performance of various machine learning models on the Named Entity Recognition task, an important challenge in the realm of Natural Language Processing. The task involves correctly identifying and labelling named entities, such as persons, organizations and locations in text. The study evaluates four models: Logistic Regression, Naive Bayes, Support Vector Machines and a fine-tuned BERT model. The results demonstrate that the pre-trained transformer performed the best, with a micro F1-score of 0.91. The Logistic Regression and Support Vector Machines also exhibit high performance when applying a combination of word embeddings and one-hot encoded features. This research employs a token-based evaluation without incorporating the 'B-' and 'I-' prefixes of NE labels. Future research could utilize span-based evaluation methods, investigate other machine learning algorithms and feature combinations to further enhance NER performance results.

1 Introduction

Natural Language Processing (NLP) includes the task of Named Entity Recognition (NER), which aims to identify and categorise named entities (NEs), such as persons, locations and organizations in a text (Nadeau and Sekine, 2007). In this study, various machine learning (ML) algorithms including Logistic Regression (LR), Naive Bayes (NB), and Support Vector Machines (SVM), and a fine-tuned transformer model (BERT) are applied to the NER task. One-hot encoded features and word embeddings are employed in isolation and in combination to train and test the models.

The evaluation results show that with one-hot encoded features the SVM model achieves a micro F1-score of 0.82, while LR and Multinomial NB systems achieve F1-scores of 0.80 and 0.78 respectively. Incorporating word embeddings alongside one-hot encoded features results in an increased performance of the SVM classifier (micro F1-score 0.87), the hyper-parameters of this system were fine-tuned for the NER task. The pre-trained and fine-tuned BERT model reaches the highest F1-score of 0.91 for this task.

2 Related work

In early stages, handcrafted rule-based models with extensive rule engineering were prominent for the NER task, which depend on a set of rules to identify NEs. While these models are straightforward and easily interpretable, their performance is limited (Nadeau and Sekine, 2007). Later on ML algorithms, such as SVMs, Conditional Random Fields and Decision Trees were applied. These systems utilise supervised learning to treat NER as a statistical classification task. By learning conditional probabilities from annotated training data, the systems become more robust compared to rule-based models. Hybrid systems have emerged, combining rule-based and machine learning-based approaches, which deliver improved results (Mansouri et al., 2008). In recent years, deep learning models, which leverage neural networks and word embeddings for contextual information, have become state-of-the-art approaches for NER (Shen et al., 2017).

NER systems employ a range of features to effectively classify NEs in text. Orthographic features, such as capitalization patterns and punctuation, provide insights into word identity (Mansouri et al., 2008). Syntactic features, including part-of-speech (POS) tags, chunk tags and morphological information, assist the model in understanding the grammatical roles of tokens and facilitating correct classification. Some papers mention the concept of a "word window", which allows the model to capture information about the preceding and following tokens' identity using word embeddings. Gazetteer features, which involve matching tokens against lists and dictionaries, are useful to identify

abbreviations and rare NEs. A combination of features are often used to improve efficiency of NER systems, which is determined by the available data and characteristics of the utilised system (Nadeau and Sekine, 2007).

3 Task and Data

3.1 Task

Named Entity Recognition (NER) is a subtask of NLP, which aims to identify and classify NEs in textual data. The task involves extracting information about entities, such as organizations, locations, people from unstructured text. NER has an integral part in several applications, including text summarization, information extraction and machine translation. Four NE categories are distinguished in our task, these are persons (PER), organizations (ORG), locations (LOC) and miscellaneous entities (MISC). The task follows the IOB-annotation scheme, indicating whether the token is part of a NE, and what kind of NE it is. Tokens outside NE spans are labelled 'O', while tokens within NEs are labelled 'B-' for beginning or 'I-' for being inside a NE expression.
The CoNLL-2003 data set is designed for NER, providing a structured representation of labeled tokens. Each line in the data is organised into four tab-separated key categories: the token(1), the POS-tag(2), chunk tag(3), and NE label(4) of the token. Sentence boundaries are indicated by empty lines. We may assume that NEs in the data are neither recursive nor overlapping (Tjong Kim Sang and De Meulder, 2003).

3.2 Data Set and Distribution

The data comprises training set, development set and a test set. Tokens in the unprocessed training data belong to nine distinguished NE classes following the IOB-annotation scheme. Most tokens belong to the 'O' class, followed by those in classes with the 'B-' prefix, indicating the token's first position within a NE span. Fewer tokens belong to 'I-' classes compared to 'B-' classes, implying that most entities consist of a single token. The training and development data distribution changed after preprocessing, as shown in Table 1 and Table 2. The preprocessed training data was used to train the models, thus the distributional observations below refer to the cleaned data.

There is an imbalanced distribution among the NE labels in the data. 81.12% of the tokens belong to the 'O' class, which is the majority class in the preprocessed training data. Concerning the remaining NE classes, the second most represented class is 'PER' with 5.39%, while 'MISC' is the least represented with 2.23% of the instances. The distribution is proportionately similar in the development data, see Appendix A.

NE Class	Count	Percentage
O	167,400	83.16%
B-LOC	7,140	3.54%
B-PER	6,600	3.27%
B-ORG	6,321	3.14%
I-PER	4,528	2.24%
I-ORG	3,704	1.84%
B-MISC	3,438	1.70%
I-LOC	1,157	0.57%
I-MISC	1,155	0.57%
Total Labels	201,443	100%

Table 1: NE distribution before preprocessing in the training data

NE Class	Count	Percentage
O	167,394	81.12%
PER	11,128	5.39%
ORG	10,025	4.86%
LOC	8,297	4.02%
MISC	4,593	2.23%
Total Labels	206,437	100%

Table 2: NE distribution in the preprocessed training data

3.3 Preprocessing

Since the purpose of this analysis is to train and evaluate a models that can effectively identify NE labels in a token-based manner, 'B-' and 'I-' prefixes of NE labels were removed in the entire dataset, focusing only on the NE classes.

3.4 Evaluation Metrics

The evaluation could be executed either at a token or at a span level. For this analysis, a token-based evaluation is set up without the inclusion of the 'B-' and 'I-' prefixes of labels. Models are evaluated by examining the micro F1, precision and recall scores provided in the classification report. Class 'O' is omitted from classification reports, since the objective is to evaluate the models based on their capacity to correctly capture tokens in NE classes (PER, ORG, LOC, MISC). Micro-averaging is used to ensure equal weight for each prediction, irrespective of the class. By observing the confusion

matrix, we can also notice patters where the model correctly classified a NE, missed to classify a NE, or classified a NE but with the incorrect class label.

4 Models and Features

4.1 Models

Logistic Regression

Logistic Regression (LR) is a statistical modeling technique used for binary classification tasks, predicting labels based on a set of independent variables. In context of the NER task, the model is trained to predict the NE class of tokens assuming a non-linear relationship between features and labels. The model learns to assign weights to features extracted from labeled data, where each token has a corresponding NE type. After training, the model is tested to predict the NE labels of the test data.

Alternative Methods

Multinomial Naive Bayes (NB), a generative probabilistic classifier, was also implemented. The model learns the probability distribution of each one-hot encoded feature for each NE class from the labeled training data. NB calculates the posterior probability of each NE class given the set of input features and assigns the class label with the highest probability. This method assumes a conditional independence among features to execute NE classification.

Support Vector Machines (SVM) were also used for the task. SVMs are discriminative classification algorithms that can be utilized to find an optimal hyperplane that maximizes the margin between classes. In NER, SVMs learn to categorise tokens with their respective NE labels using labeled training data. SVMs aim to find an optimal decision boundary to distinguish between tokens belonging to distinct NE classes. Support vectors, which are tokens closest to the decision boundary, are used to find the optimal separating line.

Fine-tuned BERT

BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained deep-learning model, which is capable of learning contextual representations between input words. Followed by the unsupervised pre-training process with unlabelled text data, the users can fine-tune the model for specific NLP tasks with supervised learning, such as sentiment analysis or machine translation. BERT is useful for the NER task since it allows for the understanding of contextual relations to capture the correct entity type (Devlin et al., 2018).

4.2 Features

Effective NER utilises a diverse set of features that capture linguistic, orthographic, and contextual information. Five one-hot encoded features were tested and selected for training the ML algorithms. The *token* feature allows the model to associate each token's identity with its corresponding NE label. Incorporating the *previous* and *next token* features within a sentence enables the model to leverage sequential information about NE patterns to capture multi-token entities. For example, in the span 'University of Chicago', the model can predict that 'of' belongs to a specific NE class if it has information about the surrounding tokens. Integrating the *POS-tag* feature enables the model to learn connections between POS-tags with NE classes. Especially prevalent POS-tags are proper nouns ('NNP'), which constitute 60% of labeled NE tokens in the training data. Regarding orthographic characteristics, capitalization patters are informative since 69% of NE-labeled tokens have the first character capitalised. The implemented *word shape* complex feature assigns a symbolic representation to the characters of the current token by categorising them as 'd' for digits, 'c' for lowercase letters, 'C' for uppercase letters and 'o' for other characters. This complex feature analyses the orthographic structure of the token in one step, for example the word 'Ltd.' is represented as 'Cccp' for the model.

In addition to one-hot encoded features *word embeddings*, which capture semantic relationships, were also implemented. Word embeddings represent words as dense vectors in a high-dimensional space, enabling the model to capture contextual relationships between words. The system leverages the *current*, *previous* and *next token's* word embedding within an input sentence to enhance NE recognition. If the token is the first word in a sentence, i.e. it is followed by an empty line indicating sentence boundary, a zero vector is used for the previous token. Similarly, if the token is the last word in a sentence, meaning an empty line follows, the next token is a zero vector word embedding. The word embeddings of the previous, current and next token are concatenated into a single vector for training some of the models (LR,

SVMs). The combination of one-hot encoded features and word embeddings has proven to further enhance the models' performance. In instances where word embeddings were applied, the token, preceding token and next token features were excluded to reduce vector space and avoid overlapping input features.

5 Experiments and Results

5.1 Feature Ablation

Feature ablation analysis was performed to assess the impact of individual one-hot encoded features on the Logistic Regression model. I decided to complete feature ablation on this system due to its quick implementation and clear interpretability. The outcomes convinced me to exclude some features, such as chunk-tag and previous token's NE label since they did not improve the model's performance. The LR model achieves a micro F1-score 0.613 when using only the token feature. Introducing the word shape feature, which captures orthographic information like capitalization patterns and digits, results in a micro F1-score of 0.744. Incorporating the previous token feature pushes the F1-score to 0.773, and adding the next token features results in the micro F1-score 0.799, suggesting that these features enhance the model's capacity to grasp relationships between adjacent words. Adding the POS-tag feature, which encapsulates syntactic information about the token, increases the model's micro recall score to 0.794, demonstrating the significance of syntactic information for NER. The combination of these features results in an overall improvement in the model's performance, see Table 3.

Feature Combination	Micro Precision	Micro Recall	Micro F1-Score
token	0.854	0.479	0.613
token, word_shape	0.744	0.745	0.744
token, word_shape, prev_token	0.775	0.770	0.773
token, word_shape, prev_token, next_token	0.817	0.782	0.799
token, word_shape, prev_token, next_token, POS-tag	0.812	0.794	0.803

Table 3: Feature ablation analysis results for LR model

5.2 Evaluation

One-hot encoded features

The LR model with a single one-hot encoded token feature serves as baseline for the analysis, reaching the F1-score 0.613. As anticipated due to the class imbalance, the model only excels at identifying 'O' class tokens, but struggles when classifying tokens belonging to other NE classes, indicated by the low recall score. Incorporating additional one-hot encoded features (word shape, previous token, next token, POS-tag) enhances the LR model's performance, as reflected in a micro F1-score improvement reaching 0.803. The significant increase in recall suggests that the model improved in correctly classifying tokens belonging to NE classes.

Model	Micro Precision	Mirco Recall	Micro F1
LR (baseline)	0.854	0.479	0.613
LR	0.812	0.794	0.803
Multinomial NB	0.759	0.792	0.775
SVM (linear kernel, C=1.0)	0.825	0.814	0.819

Table 4: Performance results with five one-hot encoded features

The three ML models using five one-hot encoded features achieved a relatively high micro F1-score above 0.78. Notably the SVM model outperformed the LR and MultinomialNB models, as shown in Table 4. Considering precision, the Multinomial NB has the lowest score, indicating its weakness in correctly classifying tokens as NEs. The SVM classifier had the highest recall score, which highlights it is less likely to miss an entity. The confusion matrices revealed that the models were prone to misclassify 'ORG' tokens as 'PER' tokens. Multinomial NB showed the tendency to misclassify 'O' instances as 'PER' compared to the other models, see Appendix B for more details.

Word embeddings

In order to introduce semantic information and contextual relationships, word embeddings derived from the Google News pre-trained model are implemented for LR and SVM. For each sentence, a vector is created by concatenating word embeddings of the current, previous and next token within a sentence. The micro F1-score for LR slightly improves, but it remains similar for SVM compared to using five one-hot encoded features, see Table 5.

Model	Micro Precision	Micro Recall	Micro F1
Logistic Regression	0.832	0.813	0.823
SVM (linear kernel, C=1.0)	0.829	0.808	0.818

Table 5: Performance results with word embedding of current, previous and next token

Word embeddings with one-hot encoded features

To further enhance the feature representation, the

one-hot encoded features of POS-tag and word shape, are combined with word embeddings of the current, previous and next token for training the SVM and LR model. This refined feature combination results in a substantial increase in micro recall scores for both models, 0.872 and 0.847 respectively. The micro precision score of the SVM model reaches 0.877, which is a significant development compared to the previously described ML models, as displayed in Table 6.

Model	Micro Precision	Mirco Recall	Micro F1
Logistic Regression	0.835	0.847	0.841
SVM (RBF kernel, C=10)	0.877	0.872	0.874

Table 6: Performance results with word embeddings and one-hot encoded features

Hyper-parameter Tuning

To optimise the hyper-parameters of the SVM model combining word embeddings and one-hot encoded features, Random Search was employed to find the best parameters, including kernel type and C parameter. This method efficiently explores the various hyper-parameter combinations through random sampling and evaluates performance using a five-fold cross-validation process with eight iterations. A train-test split was implemented to systematically identify the optimal set of parameters with the maximum F1-score. This process returned the best hyper-parameters for using the SVM model with the combination of word embeddings and one-hot encoded features[1]. The training and development data were used for this process, the best model (RBF kernel, C=10, gamma=scaled) was tested on the test data, see results shown in the previous subsection.

Fine-tuned BERT

Using the a fine-tuned BERT model results in the highest performance over the previously discussed systems. It is noticeable that precision and recall scores are relatively in balance for each class, whereas other systems often show an imbalance between these scores, see Table 7. The performance is higher compared to previous models due to the characteristics of BERT, which includes utilising contextual embeddings, bidirectional attention and pre-trained knowledge, thus enabling the transformer to grasp language patterns and contextual information to classify NEs.

[1]Seach parameters: kernel (linear, RBF), C (0.001, 0.01, 1, 10), gamma (scaled)

Class	Precision	Recall	F1-score
LOC	0.92	0.93	0.92
MISC	0.77	0.78	0.78
ORG	0.89	0.89	0.89
PER	0.96	0.96	0.96
Micro avg	0.90	0.91	0.91

Table 7: Classification report for the fine-tuned BERT

6 Error Analysis

6.1 General findings

I decided to analyse the best-performing supervised ML system, the SVM model combining word embeddings and one-hot encoded features with optimized hyper-parameters to explore options for further improvement. The system achieves a micro F1-score 0.874 and demonstrates the ability to correctly predict 'PER' NEs with only a few misclassifications, which is the second largest NE class after 'O'. The model mislcassifies 1343 tokens, 21% are multi-word NEs and 79% are single-word NEs. The confusion matrix reveals the model's tendency to misclassify 'ORG' tokens as 'LOC' tokens, which occurs 215 times. This applies in both directions, the system tends to misclassify 'LOC' tokens as 'ORG' ones (120 instances). Another issue is that the model often confuses the 'ORG' and 'O' tokens. There are more than 143 instances when the gold label was 'O' but the model predicts 'ORG', and 159 cases where the true label is 'ORG' and the model predicts 'O', as shown in Figure 1.

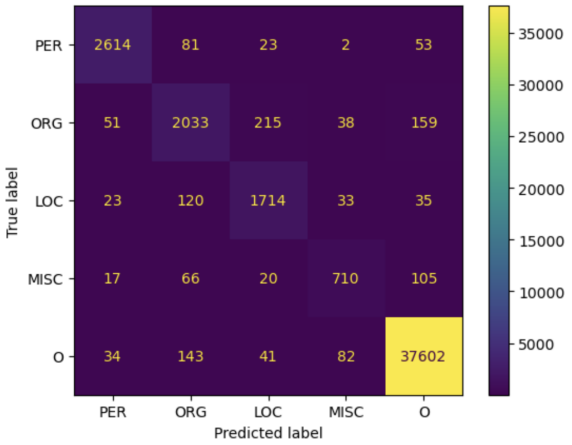

Figure 1: Confusion Matrix for SVM (RBF kernel, C=10) with word embeddings and one-hot encoded features

6.2 Confusing organisations and locations

A significant error pattern is the misclassification of 'ORG' tokens as 'LOC' tokens and vice versa. Errors of this kind belong to three main groups. The issue is prevalent for sports teams, such as 'CHICAGO','NEW JERSEY' and 'Utrecht', corporate entities like 'Bre-X' and miscellaneous organisations like 'Animal Hospital', 'Catholic Church'. Likewise, the instances where the system misclassified 'LOC' tokens as 'ORG' ones include geographical locations, such as 'Syria' and 'Budapest', sports-related locations like 'Melbourne Cricket Ground', all uppercase locations like 'DALLAS' and first elements of location names like 'St.', 'West'. It is reasonable that the model is struggling with distinguishing between organization and location names, especially since they can often be ambiguous or overlapping as the examples show. However, several counterexamples can be found, where the model correctly classifies tokens of similar nature, such as 'RUGBY UNION' (ORG), 'Irish Republican Army' (ORG), 'British Airways' (ORG), 'U.S. Supreme Court' (ORG), U.S. (LOC), GLASGOW (LOC).

6.3 Confusing organisations and 'O' labels

Another weakness of the model is the misclassification of 'ORG' labels with 'O' labels and the other way around. Firstly, cases where the gold label is 'ORG' but the model misses to identify the token as an entity follow certain patterns. It frequently occurs in case of uppercase sports organisations, such as 'EAGLES' and 'ARSENAL'; sports or business-related tokens like 'League', 'Finance', 'Trading'; sports teams' names that include numeric values, such as 'SV 1860 Munich' and 'Austria III'; abbreviated organisation names like 'NYMEX' and 'ILO'; and function words in multi-token NE spans, such as 'of' and 'and'. Furthermore, there were 149 instances where the gold label was 'O' but the model assigned the organisation label to the token. These cases were primarily due to acronyms that resemble organisation names, such as 'GUNMEN' and 'NFC', and words similar to political organisations like 'Congress', 'Ministry' and 'Government'. However, there are several examples where the model correctly predicts the organisations of similar kind, for instance 'St. Pauli' (ORG), 'II Tilburg' (ORG), 'Department of Transport' (ORG), 'West Hartlepool' (ORG).

7 Discussion

The limitations of this study should be acknowledged. The imbalanced distribution of NE labels, particularly the prominence of the 'O' class, introduces biases into the model's prediction capabilities. For this reason, class imbalance could be addressed in future work to enhance the models' performance in predicting minority classes. The analysis was conducted under time constraints, which impacted the scope of experimentation. Time permitting, I would be interested in exploring more machine learning algorithms, trying different feature combinations and conducting more extensive fine-tuning. Additionally, this analysis employs a token-based evaluation without incorporating the 'B-' and 'I-' prefixes of NE labels. Preserving the prefixes in the data and using a span-based evaluation could provide a deeper insight into the models' performance. Investigating a wider selection of features and evaluating feature interactions could be a subject of future research.

8 Conclusion

In summary, the study explores the performance of various machine learning models, including Logistic Regression (LR), Multinomial Naive Bayes (NB), Support Vector Machines (SVM), and a fine-tuned transformer model (BERT), for the Named Entity Recognition (NER) task using the CoNLL-2003 dataset. Various feature combinations were tested and fine-tuning of the SVM classifier was conducted using the development set, the final results in this report reflect the models' performance on the test set. Based on the evaluation scores, the fine-tuned BERT model reaches the best results with a micro F1-score of 0.91. The SVM classifier combining word embeddings and traditional features follows closely, reaching 0.88, with a computationally demanding training time. The error analysis provides insights into this model's strengths and weaknesses, showing a need for a more in-depth examination of possible feature interactions. We may conclude that incorporating word embeddings significantly improves the performance of the LR and SVM models. One-hot encoded features however, still hold value, as some of the best results were achieved by combining word embeddings with one-hot vectors.

Acknowledgements

Throughout the semester I collaborated with my buddy Murat Ertaş, we developed some parts of code together, especially the code for extracting and combining word embeddings. Discussions with Christina Karavida inspired me to explore the idea of advanced features, especially 'word shape'. I provided help for Szabolcs Pál with putting the code together.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Oliver Kramer. 2013. K-nearest neighbors. In *Dimensionality Reduction with Unsupervised Nearest Neighbors*, volume 51 of *Intelligent Systems Reference Library*, chapter 2. Springer, Berlin, Heidelberg.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4):885–916.

Alireza Mansouri, Lilly Suriani Affendey, and Ali Mamat. 2008. Named entity recognition approaches. *Faculty of Computer Science & Information Technology, University Putra Malaysia*.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Preslav Nakov, Zornitsa Kozareva, Alan Ritter, Sara Rosenthal, Veselin Stoyanov, and Theresa Wilson. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. *Second Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 312–320.

Yanyao Shen, Yakov Kronrod, Hyokun Yun, Zachary C. Lipton, and Animashree Anandkumar. 2017. Deep active learning for named entity recognition.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, Edmonton, Canada. Association for Computational Linguistics.

Zhongheng Zhang. 2016. Introduction to machine learning: k-nearest neighbors. *Ann Transl Med*, 4(11):218.

Theoretical Component

What is Machine Learning?

Machine Learning is the process whereby computers learn from a large amount of labelled or unlabelled data (i.e. experience) in a supervised, semi-supervised or unsupervised manner to carry out a specific task or a series of tasks with the best possible performance without any explicit programming. The type of data may include, but is not limited to, images, texts, audio and structured data. This data is divided into training, development and test data, the former two are used to iteratively improve the model, the latter to measure the model's performance.

The quality of data can influence the machine learning models' performance on the task, preprocessing is a crucial step to clean the noisy data before presenting it to the system. Feature engineering is another key aspect of the process, whereby data is structured into a numerical representation that is comprehensible for the system. While working with machine learning systems, one should also consider the ethical implications and consequences.

Sentiment Analysis in Twitter

1.A.1 Subtask B

Sentiment analysis is a task within NLP with the objective of determining the sentiment polarity of textual data. Subtask B is a multi-class classification task, which focuses on processing tweets and SMS messages (input) and categorize them as positive, negative or neutral (output) based on their sentiment on a message-level using a trained classifier. If the text message contains mixed sentiment, the dominant sentiment should be classified (Nakov et al., 2013).

1.A.2 Overview of features

For the task of identifying sentiment of a message a range of features extracted from the text could be useful. Using word-specific features from the text, such as n-grams, word clusters and word embeddings allows us to capture words that carry sentiment and understand them in contextual relation to one another. For example, recognizing the positive sentiment in *'amazing'* and capturing its intensity by the modifier adjective in *'amazingly beautiful'*. Orthographic features, including capitalization and punctuation marks, highlight the intensity and dominance of a particular token in the message, for example *'INSANE!!'* carries more weight than *'insane'*. Repeated characters often enhance the importance of a word, like ' *booooring*'. Negation words change the polarity of the following words in a sentence or clause, such as the word *'not'* in the sentence *'She was not impressed, but said the film was fine.'* Syntactic features, including POS-tags, help disambiguate the message and identifying the words that carry sentimental meaning, such as adjectives and adverbs. Dependency relations allow us to identify modifiers in the message, which often change the sentiment of sentence heads, for example *'Even with a bit of trouble, we nailed the project lunch'* has a positive sentiment. Social-media specific features, for instance emoticons, commonly used abbreviations like *'lmao'* and hashtags like *'#metoo'*, *'#socialismsucks'* convey informal language often loaded with sentiment. Sentiment lexicons, such as *SentiWordNet* or *AFINN* enhance the model's capacity to identify sentiment by incorporating sentiment scores from the lexicon. Tone markers can also help identify the subtle forms of sentiment, such as irony, sarcasm or humour, for example *'This was the last thing I needed today'* could either carry positive or negative sentiment depending on the context.

1.A.3 Representation of features

Human readers have the innate ability to understand words in context, identify words that express emotion, and grasp subtle nuances in meaning. Computational algorithms on the other hand, require features to be represented in numerical forms, such as one-hot encoding or bag-of-words (BOW). The chosen representation impacts the way the model processes textual input, which leads to trade-offs in terms of complexity and interpretability. For the task of sentiment analysis, lemmatizing the text and representing tokens as BOWs is a viable option since it reduces the sparsity of vector length, but it also comes at the price of losing word order. Representing tokens with one-hot encoding however, preserves word order but also increases vector length by capturing the presence or absence of each token in the text.

One of the most informative features is the presence or absence of a token in a sentiment lexicon. A potential implementation of this feature involves the system checking each unique token against the sentiment lexicon. If it is in the lexicon, the corresponding sentiment score could be added to the

BOW representation of the token. Tokens not found in the lexicon could retain their original representation. Another meaningful feature is the casing of a token, which indicates whether the token carries more weight than others. One solution involves creating two separate BOWs, one for lowercase tokens and another for uppercase tokens. Concatenating these representations enables the machine to learn to assign more weight to uppercase words. Special considerations must be made to avoid assigning additional weight to single-character tokens or tokens in the first position of a sentence (e.g. personal pronoun *'I'* or 'A' as first word of a sentence).

Emoticons are frequently used in social media to convey emotional nuances within text, representing these emoticons can assist the model capture the sentiment of the text. A binary indicator could be created for each unique emoticon in order to represent its presence or absence in the text. For instance, if the text contains a crying face emoticon, its corresponding binary indicator would be set to 1; otherwise, it would be 0. If a BOW representation is employed, the binary emoticon indicators could be integrated by concatenating them with the BOW vector. Since there is a large variation between emoticons, such as multiple versions of laughing emojis, similar emoticons could be grouped together to minimize redundancy and ensure a comprehensive representation. It is essential to highlight that the combination of these and several other features is required to improve the performance of the model for sentiment analysis.

Deterministic Coreference Resolution

1.B.1 Task definition

The task of coreference resolution (CR) involves identifying and grouping all anaphoric mentions within an input text that refer to the same entity (Lee et al., 2013). A mention is considered anaphoric if it refers back to an entity previously mentioned in the text. The task extends beyond sentences boundaries, allowing for the identification of entities across multiple sentences. The system takes an input text and returns a set of clusters after processing, where each cluster is composed of all mentions referring to a common named entity. For example, in the text input 'Sarah went to the party. She had fun, her brother also came.', the pronoun 'she' is an anaphoric mention referring back to the entity 'Sarah'. Consequently, the resulting output cluster should include 'Sara', 'she' and 'her'.

CR is a challenging task since it requires information about semantic, syntactic and contextual relationships between input words. In order to identify anaphoric mentions, the system must recognise not only pronouns and noun phrases, but also expressions implicitly referring to a named entity. Furthermore, the system must distinguish whether the phrases point to the same entity or a different one, which can be achieved through an understanding of semantic connections and utilization of linguistic information.

1.B.2 Overview of features

In order to solve the complex CR task, not only entity information, but also linguistic, semantic and discourse features can be useful for creating an optimal the system. Named entities are crucial for this task, for example *'Anna'* in the sentences '*Anna went to the shop. She bought flowers.*' is a named entity with the feature label PERSON. Speaker identification is utilized to find pronouns, such as *'I'*, *'me'*, *'mine'*, that refer to the speaker of the sentence and assign them a feature. These two features help the model conclude that *'she'* is likely to refer to*'Anna'* in the example sentence. The feature string match allows the model to capture references between speaker names and occupations that might be referents of the same entity, for example to identify the connection between *'Dr. White'* and *'doctor'*.

Among the linguistic features, POS-tags, more specifically pronouns and noun phrases, are useful to identify if a certain token is a potential entity. For instance, the pronouns*'he'*, *'him'*, *'she'*, *'her'* frequently refer to persons previously introduced in the text. The feature of morphological numbers allows the model to identify the number (singular or plural) and grammatical gender (masculine, feminine, neutral) of the entity. This feature is especially useful to decide if two mentions in the same context refer to the same or different entity since the morphological features of the co-referents is expected to be the same. For example in the input text *'John saw a dog outside. It was walking on the street'*, the morphological number feature (singular) and the gender feature (neutral) assist the system to form a coreference cluster for the mentions *'dog'* and *'it'*. Dependency parsing is a useful tool to extract the head match feature inside the sen-

tence, which can be utilised to precisely determine coreference. In the example *'Anna is a teacher. She works at a school'*, the system can identify a match through dependency parsing and correctly assign the noun phrase heads to one cluster.

Considering orthographic and semantic features, acronyms are informative features to establish if two proper nouns, such as *'EU'* and *'European Union'* are referents of the same entity. Additionally, a feature indicating the semantic distance between two tokens can be used to identify semantic links between phrases, such as *'superstar'* and *'actress'*. A further feature from the discourse perspective is the relative distance between two entities, which can be used to measure if two mentions are close enough in the text to be clustered together. The feature of animacy distinguishes between animate and inanimate mentions, which is informative since most entities are animate. In the example *'The dog hit the ball, it barked.*, the pronoun *'it'* refers to *'the dog'* since they share animacy.

The rule-based system introduced in the paper is based on a series of applicable rules that identify and filter entities. These rules are in a hierarchical structure of sieves, where more general rules are applied first in order to reach high precision, then specific sieves are used to filter the potential entities and improve recall. The first step in the hierarchy is mention detection; in this step the system finds all phrases that refer to an entity, which is followed by ten sieves and post-processing. First, the system selects phrases that are likely to be mentions, such as pronouns *('she', 'her')*, definite noun phrases (*'the woman'*) and proper nouns (*'president Bush'*). This step is executed with the assistance of POS-tags and head dependency relationships. The sieves could be categorised into groups, in order to illustrate the representation of the required features, this analysis focuses on sieves 2-3 and sieve 10 below.

1.B.3 Representation of features

Sieves 2 and 3 in the CR system involve the application of strict and relaxed string match methods, which are useful to identify matches that are exactly or partially the same. Exact string matches are for example the noun phrases in *[the Shahab 3 ground-ground missile]* and *[the Shahab 3 ground-ground missile]*. Relaxed string matches are approximate mention matches, such as *[Clinton]* and *[Clinton, whose term ends in January]*. In the first step the system compares entire strings character by character to identify exact string matches. The second step involves comparing strings obtained by disregarding tokens after head words, such as post-modifiers and relative clauses. Key features for this step include the possibly one-hot encoded POS-tag feature to identify noun phrases. Consequently, the system tokenizes the selected noun phrases, and compares the tokens to establish the presence or absence of overlap. For each pair of mentions, head matches could be represented as binary features; indicating whether the mentions share the same head. Depending on the degree of overlap between the tokens and dependency heads, a cosine similarity score could be assigned to each pair of noun phrases. This measure facilitates the identification of matching strings above a defined score threshold.

Sieve 10 is the final sieve before preprocessing, which focuses on pronominal coreference resolution to link pronouns with their corresponding antecedents, such as 'John' and 'he' in the example *'[John] is a musician. [He] played a new song'*. Indicative features for this process involve grammatical agreement features, such as number (singular or plural), gender (feminine, masculine and neutral), animacy (animate or inanimate) and person (first, second, or third) of the pronoun and the antecedent, which could be extracted using lexicons and dictionaries. NER labels could help identify if the pronouns and and antecedents are of the same NE class. Moreover, the feature of word distance, meaning the number of tokens separating the target words, can be used to limit the search space between pronouns and potential antecedents. The listed features could be represented as a combination of binary representations, such as number or animacy, and numerical vector representations, for example word distance, to capture the presence or absence of features and their values.

K-nearest neighbors

The K-nearest neighbor (KNN) is an instance-based and non-parametric, supervised learning algorithm, which can be used for classification and regression tasks. The algorithm operates by assigning a class label to a new data point based on the class label of the majority class among its K-nearest neighbors in the training data. The primary assumption behind this method is that data points close to each other are likely to have the same class label.

The term non-parametric means that the algorithm makes no assumptions about the underlying data distribution, whereas instance-based means that the model uses the training data directly to complete classifications or predictions. The algorithm is often referred to as 'lazy learning model' because it defers the computation of the classification or prediction until a new data point is emerging (Kramer, 2013).

The functionality of the KNN algorithm can be described in the following steps: (1) calculating the distance between the query point and all other data points in the training data using Euclidean distance, (2) select the K-nearest neighbors based on the calculated distances, (3) assign the class label of the majority class among the K-nearest neighbors to the query point. Using the KNN classifier has the advantages of easy application, flexible adaptation to new data, and only a few parameters are required. However, it might not be well scalable to large datasets, have the tendency to be sensitive to outliers and noise in the data and might be computationally expensive. The KNN classifier may be applied for various tasks, including pattern recognition, image classification and fraud detection (Zhang, 2016).

NER-research preparation

Precision, Recall and F-score

Standard metrics, including accuracy, precision, recall and F-score can help us understand the performance of the model on the test data. After testing our model, each instance can fall into one of these four categories: (1) true positives, (2) true negatives, (3) false positives and (4) false negatives.

Precision is a numeric value, which indicates the ratio of correct predictions and the total number of predictions by the model.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{1}$$

Recall is a numeric value, which indicates the ratio of correct predictions and the total number of predictions the model should have found.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2}$$

F1-score is the harmonic mean of precision of recall, which combines them into a single score.

$$\text{F-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

A *high recall* is desirable if we aim to minimize the number of false positives, which comes at the cost of many false positives. An example for this is a tsunami alerting system in high-risk countries, in which case there is a high risk for missing out on the alarm, but there is a low risk for acting. Missing out on an alarm could be catastrophic, but warning people and not having an actual dangerous situation causes no harm.

A *high precision* is desirable if we aim to minimize the number of false positives, which comes at the cost of many false negatives. An example for this is a model that can detect spam emails, in which case there is a low risk of missing out, but a high benefit for acting. If the user never wants to look into the spam folder, we want to make sure that we minimise the chance that a non-spam email is ever placed into the spam folder, but this also means that sometimes spams might land in the main inbox.

There is always a trade-off between precision and recall in our model's performance, therefore, it is essential to establish what we wish to optimize the model for. Tuning the model for higher precision will come at the cost of lower recall, and the same is true for the opposite.

Token-based Evaluation

A span-based evaluation would be able to provide complex and accurate predictions about the NER task by considering the entire entity as a unit. It allows us to evaluate the model based on its capacity to identify the starting and final token of each entity span. On the other hand, a token-based evaluation considers each token individually, meaning it does not consider the context of tokens in the data set. Evaluation on the token level is more straightforward, since the model is not required to keep track of the beginning and end of each entity, which reduces complexity and the chance of errors. The disadvantage of this approach is that the system does not penalize the model for incorrectly identifying entity boundaries, which means there is information loss. For my analytical purposes a token-based approach is applied, without the consideration of 'B-' and 'I-' prefixes for simplicity, flexibility and a more fine-grained approach.

Time spent

Week	Task	Time
1	Watch lectures	3 hours
1	Read assignment1	30 mins
1	Theoretical components	1 hour
1	Explore dataset	2 hours
1	Explore features	4 hours
1	Watch lectures	5 hours
1	Math foundations	1.5 hours
1	Setting up evaluation	3 hours
2	Describe the task	2 hours
2	Preprocessing steps	2 hours
2	Implement basic system	4 hours
2	Code for evaluation	4 hours
2	Describe results	3 hours
2	Format report	2 hours
2	Watch lectures	3 hours
2	Andrew Ng Course	3 hours
3	Watch lectures	3 hours
3	Theoretical component	6 hours
3	Feature extension	2 hours
3	New ML methods	4 hours
3	Code word embeddings	4 hours
3	Literature review	2 hours
4	Hyperparameter tuning	4 hours
4	Literature review	2 hours
4	Watch lectures	2 hours
4	Revise report	4 hours
4	Extend report	5 hours
5	Theoretical component	2 hours
5	Advanced features	2 hours
5	Feature ablation	4 hours
5	Running more advanced models	4 hours
5	Updating report	6 hours
5	Watch lectures	2 hours
5	Reading material	5 hours
6	Error analysis	8 hours
6	Watch lectures	2 hours
6	Reading material	3 hours
6	Update report	5 hours
6	Code cleanup	8 hours
6	Intro & Abstract	2 hours
6	Discussion & Conclusion	3 hours
7	Exam preparation	30 hours
7	Refining report	5 hours
Total		150 hours

Table 8: Time overview.

A Additional results

	PER	ORG	LOC	MISC	O	Total
Training	11,128	10,025	8,297	4,593	169,578	203,621
Development	3,145	2,092	2,094	1,268	42,759	51,358
Total	14,273	12,117	10,391	5,861	212,337	254,979

Table 9: NE distribution in the preprocessed training and development data

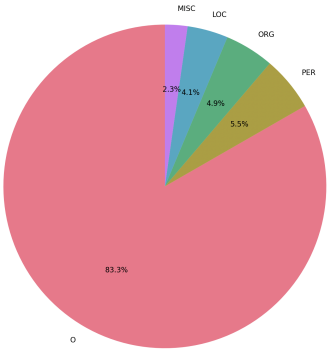

Figure 2: Distribution of NE labels in the preprocessed training data

No.	PER	ORG	LOC	MISC
1	Clinton	of	U.S.	Russian
2	Mark	Reuters	Germany	Cup
3	Michael	Newsroom	Britain	German
4	Paul	Inc	Australia	Open
5	John	St	England	French
6	David	New	France	American
7	M.	Corp	Spain	European
8	Dole	Party	Italy	Australian
9	Arafat	United	New	Iraqi
10	Wasim	National	LONDON	Bank

Table 10: 10 most frequent NE tokens per NE Class in the preprocessed training data

B Classification Reports

Class	Precision	Recall	F1-score
Logistic Regression			
PER	0.806	0.914	0.856
ORG	0.793	0.722	0.756
LOC	0.850	0.756	0.800
MISC	0.808	0.710	0.756
Micro avg	0.812	0.794	0.803
Multinomial NB			
PER	0.839	0.875	0.856
ORG	0.703	0.740	0.721
LOC	0.701	0.814	0.753
MISC	0.819	0.636	0.716
Micro avg	0.759	0.792	0.775
SVM (linear kernel, C=1.0)			
PER	0.835	0.916	0.874
ORG	0.808	0.741	0.773
LOC	0.844	0.796	0.819
MISC	0.793	0.741	0.766
Micro avg	0.825	0.814	0.819

Table 11: Classification reports for models with five one-hot encoded features

Class	Precision	Recall	F1-score
Logistic Regression			
PER	0.929	0.878	0.903
ORG	0.781	0.757	0.769
LOC	0.832	0.837	0.834
MISC	0.697	0.720	0.708
Micro avg	0.832	0.813	0.823
SVM (linear kernel, C=1.0)			
PER	0.938	0.857	0.896
ORG	0.792	0.753	0.772
LOC	0.809	0.851	0.830
MISC	0.677	0.716	0.696
Micro avg	0.829	0.808	0.818

Table 12: Classification reports for models with word embeddings of previous, current, and next token

Class	Precision	Recall	F1-score
Logistic Regression			
PER	0.938	0.933	0.935
ORG	0.783	0.786	0.784
LOC	0.833	0.851	0.842
MISC	0.685	0.745	0.714
Micro avg	0.835	0.847	0.841
SVM (RBF kernel, C=10)			
PER	0.954	0.943	0.948
ORG	0.832	0.815	0.823
LOC	0.851	0.890	0.870
MISC	0.821	0.773	0.796
Micro avg	0.877	0.872	0.874

Table 13: Classification reports for models with word embeddings combined with one-hot encoded features

C Confusion Matrices

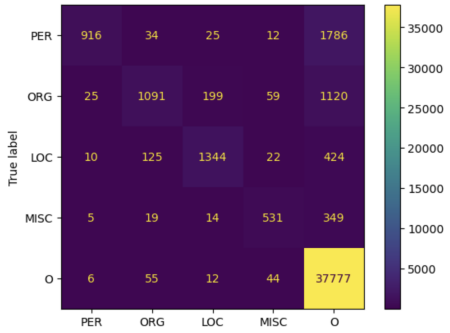

Figure 3: Confusion Matrix of the LR model with a one-hot encoded single feature (token)

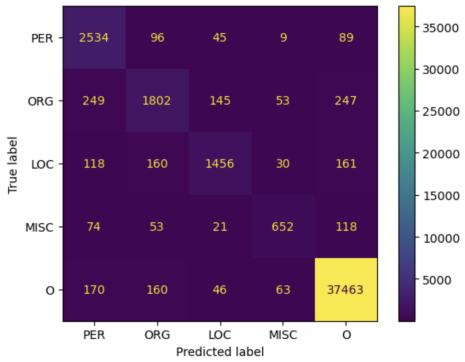

Figure 4: Confusion Matrix of the LR model with five one-hot encoded features

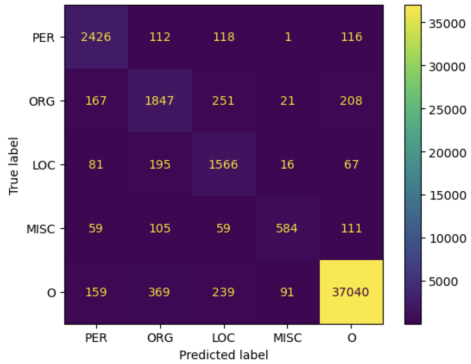

Figure 5: Confusion Matrix of the Multinomial NB model with five one-hot encoded features

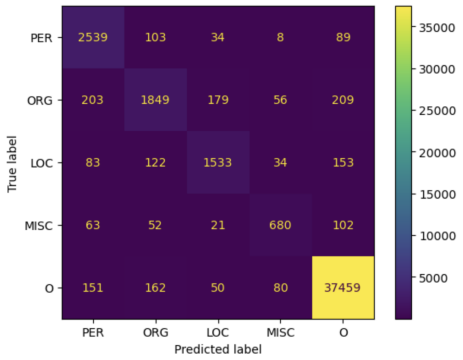

Figure 6: Confusion Matrix of the SVM model with five one-hot encoded features using linear kernel, C=1.0

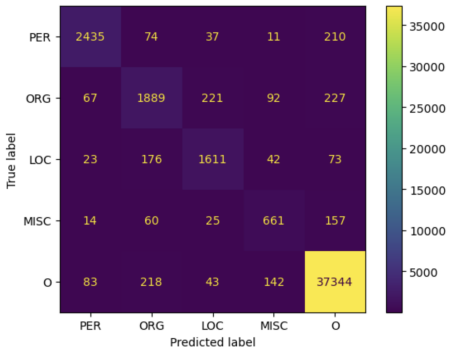

Figure 7: Confusion Matrix of the LR model with word embeddings of current, previous and next token

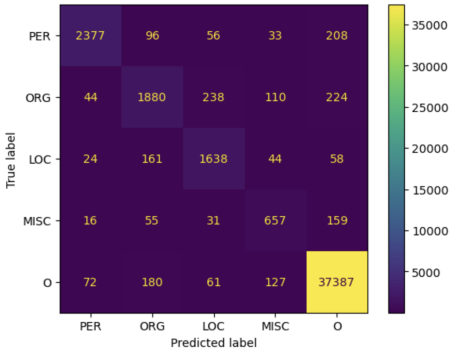

Figure 8: Confusion Matrix of the SVM model with word embeddings of current and previous token (linear kernel, C=1.0)

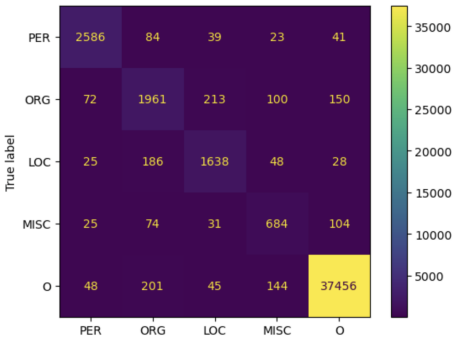

Figure 9: Confusion Matrix of the LR model with word embeddings and one-hot encoded features