Machine Learning for Named Entity Recognition and Classification

Ino van de Wouw

Abstract

Named Entity Recognition (NER) represents a critical challenge in Natural Language Processing, requiring the accurate identification and classification of entity mentions within unstructured text. This research evaluates multiple machine learning approaches for NER, comparing traditional statistical methods against modern neural architectures. Our investigation encompasses logistic regression, Support Vector Machines (SVM), Naive Bayes, and a fine-tuned BERT model, accompanied by systematic feature ablation studies examining the impact of linguistic characteristics including token morphology, context, and part-of-speech information. The experimental results demonstrate the superior performance of transformer-based approaches, with BERT achieving F1 scores of 94.6% across multiple random seeds. All code can be accessed on GitHub.

1 Introduction

Named Entity Recognition (NER) represents a fundamental challenge in Natural Language Processing, requiring systems to automatically detect and classify named mentions of entities like persons, organizations, and locations within unstructured text. This research evaluates multiple machine learning approaches for addressing the NER task, focusing on both traditional statistical methods and modern neural architectures.

Our investigation encompasses several complementary modeling strategies: logistic regression and Support Vector Machines (SVM) as baseline approaches, Naive Bayes for probabilistic classification, and a fine-tuned BERT transformer model leveraging contextual embeddings. We systematically analyze the contribution of various linguistic features through comprehensive ablation studies while exploring how different model architectures handle the complexities of entity recognition.

The experimental results demonstrate the superior performance of the BERT-based approach, achieving F1 scores of 94.6-94.7% and accuracy of 98.7% across multiple random seeds. Traditional methods showed varying degrees of effectiveness, with the standard SVM and logistic regression models reaching F1 scores of 77.0% and 75.7% respectively. Feature ablation studies revealed the critical importance of contextual information and token characteristics, particularly the impact of preceding tokens and capitalization patterns on classification accuracy.

2 Task and Data

2.1 Task

Named Entity Recognition (NER) is a crucial technology in Natural Language Processing, it enables the automatic identification and categorisation of specific elements in texts, such as person names, organisations, locations, dates, and other entities. NER is essential in transforming vast amounts of unstructured texts into structured data that can be used for analysis.

2.2 Dataset and Distribution

The English CONLL (Sang and De Meulder, 2003) dataset contains 203.621 tokens, each row represents a tokens or the end of a sentence, which is marked by an empty line. All of the tokens are labelled on the following features: PoS-Tag, Chunk-Tag and Named Entity Tag, we will focus solely on the Named Entity Tag for the distribution. The Named Entity label contains the following classifications, similar to the system proposed by Ramshaw & Marcus (1995): 'O' : Tokens that contain this token are classified as outside of a named entity. The majority of the tokens have this classification.

'B - __' : Tokens which classification begins with 'B' indicate the beginning of a Named Entity, after which a type of entity will be specified.

'I - __' : Tokens which classification begins with 'I' indicate the continuation of a Named Entity, if two separate named entities appear after each other the second will receive a 'B'.

The latter two can be followed by either of the following four types:

1. 'ORG' : Denotes that the token is of the category 'Organization'.

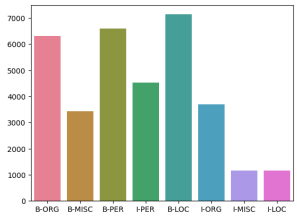

Figure 1: Named Entity label distribution in the training set, with 'outside' removed

2. 'PER' : Denotes that the token refers to a 'Person'

3. 'LOC': Denotes that the token describes a 'Location'

4. 'MISC': Denotes that the token that adhered to neither of the previous types, but were names or adjectives.

An important note is that this system of tagging makes the following assumption. All entities are non-recursive and non-overlapping, meaning that all entities can only have one category and if a named entity contains another named entity only the first is annotated.

The training data has the following distribution (figure 3 in the appendix), the majority of the labels fall in the outside category, as the training data contains sentences and does not solely exist out of Named Entities. In order to see how the different types of Named Entities are distributed we will have to look at figure 1, where 'O' is temporarily omitted.

2.3 Preprocessing

In order to do any analysis, the source data was split on new lines, which marked the end of each sentence. These were removed in order to not pollute the data. After this each line represented a token, Part-of-Speech tag, chunk tag, and Named Entity label. After this, all the occurrences of each NE label were analysed on its distribution.

2.4 Evaluation Metrics

When evaluating a classifier, it is essential to choose metrics that align with the tasks goal and the nature of the data. Recall measures the ability to correctly identify all positive instances, which is critical in applications like Named Entity Recognition where missing positives (false negatives) is costly. Precision, on the other hand, assesses the correctness of positive predictions as false positives are undesirable. The F-score provides a balanced measure by combining precision and recall, offering a single metric when both false positives and false negatives are of concern. Finally, accuracy indicates the overall correctness of predictions but can be misleading in imbalanced datasets where one class dominates. Together, these metrics provide a comprehensive evaluation of a classifier's performance (Jurafsky and Martin, 2019), helping to ensure it meets the specific needs of the task at hand. Macro averages are used in order to offset the dominant class 'O'.

3 Models and Features

3.1 Models

In our quest to find the best system for Named Entity Recognition several different models will be tested and evaluated. The first is a system that uses Logistic Regression.

3.1.1 Logistic Regression

This classifier identifies and categorizes named entities in texts. This type of model works by processing the feature vector, a vector representation of the word and the words features, and the entity and entity type it adheres to. The model calculates the likelihood for the token and its features to which entity they belong to, it does this for all of the pairs in the training data. If the model mistakes during training it is corrected by a loss function, after a lot of iterations this results in a final model with a minimized loss function indicating the optimized Logistic Regression model has been achieved(Goldberg, 2016).

3.1.2 Support Vector Machine

A Support Vector Machine (SVM) classifier first transforms a token and its features into a high-dimensional space where data-points are separated on the basis of their class labels, whether they belong to an entity, and if so to which. The SVM classifier then finds several optimal hyperplanes that separate the data-points on the vector representation of the token and its features. Maximising the margins between all the label classes. During training the SVM penalises proposed hyperplanes by the use of a hinge loss function, which increases penalisation when more entries are wrongly classified, after which a new hyperplane is proposed. This is done for a lot of iterations until the model manages to converge on

several hyperplanes discriminating all classes, with the lowest loss function possible. Eventually the SVM classifier will be able to predict the entity type on the basis of the vector representation of the token and its features(Goldberg, 2016).

3.1.3 Naive Bayes

A Naive Bayes classifier identifies and categorizes named entities in text by applying Bayes' Theorem to predict the probability that a token belongs to a specific entity type. It works by assuming that the features of each token are conditionally independent given the entity label. This assumption is foundational for the model as it allows to calculate the likelihood of a token adhering to a class based of the individual features. The model then selects the class with the highest probability given the token's features. During training the Naive Bayes classifier learns the probabilities of each feature for each entity class by counting the occurrences of feature-label pairs in the training data. The model is therefore not able to make mistakes during training and does not have a loss function in the way the Logistic Regression and SVM classifier have. When the Naive Bayes classifier receives new input it uses the learned likelihoods to calculate to which class the input belongs, calculating the probability for all different possible classes, choosing the class with the highest probability(Goldberg, 2016).

3.1.4 Fine-tuned BERT

A fine-tuned BERT model is a model that uses pre-trained contextual embeddings to find relationships between words and their context. For each token it takes into account the left and right context, this is different from traditional word embeddings that are generated independent of context. This way a word can have two different representations for the differing context it appears in. Typically BERT is used for Masked Language Modelling (MLM) or Next Sentence Prediction (NSP), but it can also be used for Named Entity Recognition. During training BERT adjusts its pre-trained weights in order to optimize to the Named Entity Recognition task. Advantages of using BERT is that it is great at capturing long range dependencies between tokens, as it processes the whole input from left-to-right and right-to-left, and that BERT does not rely on feature engineering, as has been done for the other models (Devlin et al., 2018). BERT also uses the WordPiece tokenizer, which was not necessary for the other models as the data came pre-tokenized and BERT has to process the whole text all at once and tokenize it itself. This WordPiece tokenizer is quite good at handling rare and out of vocabulary words by breaking words into subword units.

3.2 Features

The tokens in our dataset come with the following features: 1) the token itself, 2) the Part-of-Speech (PoS) to which the token adheres, and 3) the chunk tag. But due to the similarity of chunk tag and PoS tag it was decided to not include the Chunk tag. The additional features added are: the tokens lemma and the tokens shape . The token shape considers whether the token contains full capitals, all lowercase, first character capitalisation, digits, periods and hyphens. If neither of this is true the token was given the value 'Other'. The lemma was added as it was theorized that Named Entities do not have a different lemma form, meaning that if the lemma and the token differ, it for certain was not a Named Entity. The previous token was added as a feature, this was done as it was hypothesized that the linguistic or syntactic relationship between a named entity and the token that immediately precedes it might carry valuable contextual information. For one system, the Support Vector Machine classifier, word embeddings were added as a feature. Word embeddings capture semantic information about words by representing them as dense vectors in a 300 dimensional space. These embeddings capture relationships between words based on their context in larger corpora, so they have to be seen in relation to other embeddings, as the embedding itself holds an arbitrary value. The model used to find the word embeddings for the tokens in our dataset is the Google News word embeddings. Due to computational limitations it was only possible to add the information about the shape of token and the word embedding itself.

4 Experiments and Results

4.1 Hyper-parameter Tuning

Randomized Search was employed for hyper-parameter tuning, on the Logistic Regression, standard SVM and Naive Bayes models, due to its computational efficiency and effectiveness when there are a lot of parameters. Unlike Grid Search, which exhaustively evaluates all possible parameter com-

binations, Randomized Search implements a probabilistic approach by sampling from predefined parameter distributions, allowing for a more efficient exploration of the hyper-parameter space. This method proved particularly advantageous given our computational constraints and the complex parameter interactions in our models, as all models which were included in this hyper-parameter tuning, increased improved.

4.2 Feature Ablation

A feature ablation study was conducted to asses the contribution of each feature to the performance of the Logistic Regression model for the Named Entity Recognition task. This appraoch involved systematically removing individual features from the model and observing the impact on key evaluation metrics such as recall, precision, F1-score and accuracy. By analysing how performance degrades when specific features are excluded, the study provides insights into the relative importance and influence of each feature in predicting Named Entities. The baseline model includes all available features: the token, the lemma, Part-of-Speech tag, token shape, and preceding token. Table 1 shows the results of the ablation study, with each row corresponding to the models performance when a specific feature or combination of features was excluded.

4.2.1 Baseline

The baseline represents the performance of the Logistic Regression model with all the features present. The model manages to achieve a high F1-score, accuracy, recall and precision. Indicating that the complete set of features provides a well-rounded representation of the tokens.

4.2.2 Token removed

When the token is removed from the features given to the model, the model manages to perform pretty good. It barely drops in performance compared to the baseline. This could be due to the fact that the token and lemma are closely related and that the lemma functions as a safeguard.

4.2.3 Lemma removed

Just as when the token was removed, the performance barely drops when only the lemma is removed. Therefore the model was ran without both, in order to see whether these two features are really codependent and to see how much they do contribute.

4.2.4 Part-of-Speech

Excluding the PoS tag causes a small but noticeable decline in performance. The PoS tag provides grammatical context, which helps in disambiguating named entities from other tokens, such as distinguishing between a noun used as a person's name versus a common noun. The drop indicates that while PoS is useful, the model can still function reasonably well without it.

4.2.5 Token Shape

When the token shape is removed, the model's performance decreases more significantly compared to the removal of token, lemma, or PoS. Token shape encodes surface-level information such as capitalization, digits, and punctuation, which is particularly important for recognizing named entities like proper nouns (e.g., *"Mary"* or *"AMD"*). This highlights the importance of token shape in identifying entities based on their visual characteristics.

4.2.6 Previous Token

Excluding the previous token leads to a significant drop in all metrics, indicating that the context provided by the preceding word is crucial for identifying named entities. The sequential relationship between tokens helps the model recognize multi-word entities and distinguish between similar tokens that appear in different contexts. This drop underscores the importance of contextual information in NER tasks.

4.2.7 PoS and Token Shape

Removing both PoS and token shape results in a more substantial drop in recall and accuracy, though precision remains relatively high when compared to both features separately. This suggests that the model becomes more conservative, favouring precision over recall, meaning it identifies fewer entities but with greater confidence. This highlights the complementary role of PoS and token shape in improving both entity detection and classification accuracy.

4.2.8 Token and Lemma removed

When both token and lemma are removed the models performance drastically deteriorates across all metrics. These two features encompass the core lexical information necessary in order to identify entities. Note: both the hyper-parameter tuning and feature ablation study were performed on the development dataset, the evaluation was conducted on the test dataset.

Table 1: Feature Ablation results

feature left out	recall	precision	F1-score	accuracy
baseline	80.7	86.5	83.2	96.8
token	80.4	86.2	82.9	96.7
lemma	80.4	86.1	82.9	96.7
pos	80.2	86.2	82.8	96.6
shape	80.1	85.5	82.3	96.3
pos and shape	73.3	90.0	80.4	95.1
previous token	69.9	78.3	72.7	94.6
token & lemma	65.0	70.6	66.7	92.6

Table 2: Model Performance results

model	recall	precision	F1	accuracy
LogReg	75.7	78.2	76.6	94.9
NaiveBayes	63.0	78.5	67.4	92.6
SVM w/ Embeddings	64.4	67.0	65.5	93.1
SVM	77.0	76.9	76.7	95.1
BERT seed 1	95.2	94.1	94.6	98.7
BERT seed 2	95.1	94.1	94.6	98.7
BERT seed 3	95.1	94.4	94.7	98.7

4.3 Evaluation

The results of the experiment demonstrate a clear performance hierarchy between models. The BERT-based model consistently achieved superior performance in all metrics, with consistent results across different random seeds. The BERT model managed to achieve F1 scores of 94.6% twice and once 94.7%, while maintaining high accuracy, 98.7%. As can be seen in Table 2.

Among the more traditional machine learning approaches, the standard SVM and Logistic Regression models demonstrated competitive performance, achieving F1 scores of 77.0% and 75.7% respectively. Both models exhibited similar accuracy, with scores above the SVM, 95.1%, outperforming the Logistic Regression, 94.9%, classifier.

The SVM implementation that contained word embeddings showed degraded performance compared to its standard counterpart, with decreased metrics across all metrics. The F1-score of 65.5% reflects this as both recall and precision are down for the SVM with embeddings, 64.4% versus 77.0% on recall, 67.0% versus 76.9% on precision.

The Naive Bayes classifier showed moderate performance on the task, with a F1 score of 67.4% and accuracy of 92.6%, positioning itself in between the SVM model with embeddings and the Logistic Regression model. The Naive Bayes demonstrated a notable precision-recall trade-off, with a lower recall score, 63.0%, whilst having a relatively high precision score, 78.5%.

Examining just the precision-recall balance across the more traditional models, an interesting patterns is revealed. The Logistic Regression, SVM with Embeddings and Naive Bayes model all have a higher precision than recall, with the Naive Bayes containing the biggest disparity.

5 Error Analysis

The error analysis reveals distinct patterns in classification performance across different capitalization formats and internal character compositions, for the best performing classifier, the standard SVM classifier, which can be seen in table 3. In examining capitalization errors, we observe that Title-case tokens exhibited the highest error frequency (1,589 errors out of 8,698 Title case instances), representing a significant challenge for the classification model. The majority of these errors were mistakes where countries were classified as locations or locations were classified as countries.

This is followed by Upper-case tokens, which demonstrated substantial error rates (568 errors from 2,207 Upper-case instances). Lower-case tokens showed remarkable resilience to misclassification, with only 56 errors occurring across 23,835 instances, suggesting robust model performance for this category. The majority of the Upper-case mistakes were organisations that were classified as locations or as no named entity at all. The few Lower-case mistakes that were made all entailed either sports, and things related such as trophies or teams, or *and* and *of* that were inside miscellaneous named entities.

The analysis of internal character composition provides additional insights into model behaviour. Tokens containing digits demonstrated exceptional classification accuracy, with only 7 errors observed among 3,969 instances, indicating highly reliable digit-based feature recognition. The mistakes that were made regarding digits are years not being recognized as the beginning of a miscellaneous entity or numbers outside named entities being wrongly recognized as named entity. Period-containing tokens maintained strong performance with 47 errors across 2,806 instances, while hyphenated tokens showed slightly elevated error rates, 91 errors from

Table 3: Error analysis Capitalisation and Shape

		no error	error
Capitalisation	Upper	1639	568
	Lower	23779	56
	Title	7109	1589
Internal Shape	Digit	3962	7
	Period	2759	47
	Hyphen	1622	91
	Other	7666	60
Total		48536	2418

1,713 instances, the errors mainly contained references to locations or organizations which where classified as persons. The *Other* category, encompassing various special characters and symbols, maintained relatively low error rates with 60 misclassification's among 7,726 instances. The error's from the *Other* category were names such as McIntosh, McDonalds or trans-Atlantic, which featured a second capital in their token or a combination of hyphen and capital. Almost all mistakes on this *Other* category were named entities being labelled as outside a named entity.

The findings suggest that capitalization patterns, particularly Title and Upper-case formats, present the most significant challenges for the classification system. The stark contrast between Lower-case performance and other capitalization patterns indicates potential areas for model improvement, possibly through enhanced feature engineering or targeted data augmentation strategies. The robust performance across internal character compositions, especially with digit-containing tokens, suggests that these structural features serve as reliable classification indicators. This analysis provides valuable insights for future model refinements, particularly in addressing the specific challenges posed by various capitalization patterns while maintaining the strong performance observed in internal character composition handling.

6 Discussion

The feature ablation study revealed the hierarchical importance of different features in traditional models. The dramatic performance degradation observed when removing both token and lemma features (66.7% F1-score) underscores the fundamental role of lexical information in NER tasks. The notable impact of removing contextual features (without previous token: 72.7% F1-score) aligns with BERT's strong performance, suggesting that contextual information is indeed crucial for effective entity recognition. The experimental results highlight several key insights into the effectiveness of different approaches to Named Entity Recognition. The superior performance of BERT across all metrics demonstrates the significant advantages that transformer-based architectures hold over traditional machine learning approaches. This performance gap (BERT: 94.6% F1-score vs. traditional models: 65.5-76.7% F1-score) can be attributed to BERT's ability to capture contextual relationships and long-range dependencies, which are crucial for accurate named entity recognition.

An unexpected finding was the degraded performance of the SVM model with word embeddings compared to its standard counterpart. This suggests that the chosen embedding strategy may not have effectively captured the semantic relationships necessary for entity recognition, or that the dimensionality reduction required for computational feasibility may have lost crucial information.

7 Conclusion

This study has demonstrated the effectiveness of various machine learning approaches for Named Entity Recognition, while also highlighting the clear superiority of transformer-based models. The comprehensive feature ablation study and error analysis have provided valuable insights into the relative importance of different features and the specific challenges that remain in NER tasks. The research offers several key contributions: a systematic evaluation of traditional and modern approaches to NER, detailed understanding of feature importance through ablation studies, and identification of specific patterns in classification errors that can guide future improvements. The findings suggest that while traditional machine learning approaches can achieve reasonable performance, the future of NER likely lies in contextual models that can capture complex relationships between tokens.

Acknowledgements

References

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

Yoav Goldberg. 2016. A primer on neural network

models for natural language processing. *Journal of Artificial Intelligence Research*.

Dan Jurafsky and James Martin. 2019. *Speech and Language Processing*. Prentice Hall.

Erik 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.

Theoretical Component

Assignment 1

Machine learning is a field within artificial intelligence that enables computers to learn patterns from data, on which decisions and predictions can be made without explicitly being programmed to do so. It involves training models on large datasets allowing them to improve their performance over time. My initial understanding was that machine learning was simply teaching computers to learn on their own, but now I realize it's more about building systems that adapt to data and manage to generalize from it, rather than hard coded rules. This change is highlighted by ML's adaptability and data-driven learning on varying range of tasks.

Assignment 2

A Sentiment Evaluation

The goal of subtask B in the paper by Nakov et al. is message polarity classification, this means given a text, either a tweet or sms-message as input, the system must be able to classify the sentiment of the message. The range it can be classified on is Positive, Negative or Neutral. If a message conveys both positive and negative sentiments the system must determine which sentiment is stronger and then assign the corresponding label as output. The proposed features are: **Named Entities** Named entities were added as a feature by the use of a NER system specifically tuned to a Twitter corpus, this helps with the task as if you know where the source text contain named entities you can then decipher how those entities related to each other within the text. **Topic Recognition** If you know the topic a text contains you can then relate it to the named entity in its surrounding, after you know this it is possible to score the sentence on sentiment whilst mapping what the sentiment to that topic is. **Lexical features** Phrases that did not contain sentiment towards any of the chosen topics were removed from the dataset, so that the sentences remain contained at least one positive or negative word, with a score higher than 0.3 in SentiWordNet's database, this was done to prevent class imbalance. This score and the mapping to the word can then be added as a feature. **Punctuation** If a sentence contains deviating punctuation this is often a marker of sentiment, joy or anger can be expressed through repeating exclamation marks. i.e. *This was the worst service ever!! Don't go there!!!!!* in this utterance it is used to amplify that the experience was negative, whilst it is also possible to use it in the following way *He just proposed to me!!!!*, where it amplifies joy. It can be noted that just the presence of punctuation or deviating punctuation is not enough to score the sentence on, it can be used as a flag conveying that there might be something happening in that sentence. **Negation** It is very well possible that a sentence contains some sort of negation, this is tricky as it is important to know what part of the sentence is affected by the negation clause. As this might span the whole sentence or just a few words. Take the following sentence as an example: *I am not dissatisfied with this service*. in this case *not* counters the meaning of *dissatisfied*, it in itself is therefore an important marker for the sentence overall sentiment.

An example in which all of these features are incorporated is the following: *Apple's new iPhone isn't fantastic!!!* Named Entity: *Apple* and *iPhone* Topic: *iPhone* Negation: *n't* (negates the remainder of the sentence) Lexical features: *fantastic* (overwhelmingly positive - 0.8) Punctuation: *!!!*

These features can be represented in the following way. Named Entities can be represented in a vector through a binary representation, the first index marking whether there is a Named Entitiy in the sentence, the second index marking it's label (Person, Organisation and Location), a possible third index can mark the position of the named entity in the sentence representation. Topic's can represented as a one-hot vector, with each topic having its own distinct vector, this would then be added to the to the vector representation of the sentence. The lexical features can be added as vector of three dimensions representing the positive score, the negative score and the aggregate of the two previous scores, this would be added on token level with in the sentence representation. Punctuation can be represented in a binary way, 0 indicating no deviation of normal counts (i.e. total repeating punctuation count is below 3), where 1 represents an over abundance of repeating punctuation. Negation could be flagged in a similar fashion, where 0 represents not containing negation and 1 that negation is present. Both punctuation and negation would be added on the sentence representation level.

Coreference Resolution

Coreference resolution is the task of determining when two or more Named Entities in a text refer to the same entity. The goal is to identify all mentions

that refer to same entity and group them together. The input could be any sequence of sentences, an example is *Mary went to the park. She saw a dog there*, the desired output would then be a list of coreferents. In our example that would be [*Mary, She*] and [*park, there*], *dog* is not included as it has no coreferent in the preceding sentence. Possible features are: **Named Entities** To recognise which entities refer to each other in a text it is very useful to know what the named entities are, and especially when a new one begins. The named entities in a text could be represented as a list for the input text. **Gender and Plurality** In order to map corresponding entities it is important they align with their counter part on Gender and Plurality, if there is a mismatch this would signal a wrong coreferents pair. Exceptions would be due to not being able to recognize an entities gender or plurality, possible when the entity is of the organisation type. **Appositives** As Lee et al. (2013) mention appositives are two nominal mentions of the same construct, it could therefore be interesting to keep track of the appositives in a text, this would be a dictionary with one key having multiple values, as if there was an overarching person having multiple titles or roles.

Gender and Plurality can be represented in a categorical fashion. This would be a mapping of the mentions in a text on their gender, where 0 represents *Unknown*, 1 *Male*, 2 *Female*, 3 *Neutral*, where Neutral represents genderless constructs. Plurality is either 0 *singular* or 1 *plural*. Together with the dictionary approach for appositives the following example *The baker smiled to John, the delivery guy. Which he knows really from their days in kindergarten.* will have this representation: ['The Baker': 'He',1,0]['John':'the delivery guy',1,0].

Assignment 3

K-nearest neighbour (KNN)

A KNN is a machine learning algorithm that make no assumptions about the distribution of data. It clusters the data given by all attributes that are already present in the dataset. After such a cluster is made, new instances are added in where all the attributes are known, yet the cluster label the instance adheres to is not. To be able to assign a class to this new point they are plotted in space on the available information, then the algorithm looks at the closest data-points and their cluster label. If the majority of the selected amount of closest neighbours K is class

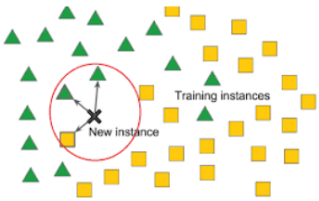

Figure 1: Example of training instances and instance to classify

Figure 2: K-Nearest Neighbour example

A, the new data-point will be assigned to class A, if the majority is class B the data-point is assigned to class B. The closest neighbour is determined by calculating the Euclidean distance between the new instance and the surrounding instances, then the K amount of neighbours with the smallest distance are selected to compare the cluster labels. If K is a small number it leads to uncertain decisions, meaning assigned cluster labels are in fact not the majority if you "zoom out", a large K makes the process computationally expensive and raises the risk for overfitting. Considering the example in in figure 3, a new data-point has been entered, first all the distances to the existing mappings are calculated. Then the K nearest neighbours are taken into account, in this case three, after which the new data-point will receive the majority label, in this example green triangle.

Time spent

Please use Table 4 to give an overview of the time you spent on each submission. You can modify the table (add new categories) if necessary.

Week	Task	Time
1	watch all three videos and make notes	1.5 hour
1	understand labels in data	1 hour
1	trying to figure out how Overleaf works	30 minutes
1	all of the coding necessary	6 hours
1	watching the four videos of week two and make notes	3 hours
1	writing a preliminary analysis for assignment 1	2 hours
2	watching all prescribed video's	6 hours
2	tinkering around with the classification script by adding features	4 hours
2	writing additional paragraph's for the report	4 hours
3	fixing word embeddings	2 hours
3	working through the BERT finetuning notebook	2 hours
3	catch-up theoretical framework as I was waiting for the fine-tuning	4 hours
3	feature ablation study (coding)	1 hours
3	feature ablation study (writing)	2 hours
3	write proper model explanations	2 hours
3	expand paragraph on features	1 hour
3	reading up on transformers and watching StatQuest	2 hours
Total	44 hours	

Table 4: Time overview.

A Example Appendix

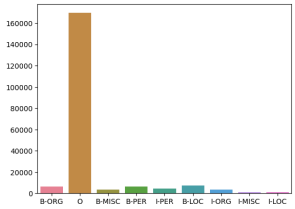

Figure 3: Named Entity label distribution in the training set

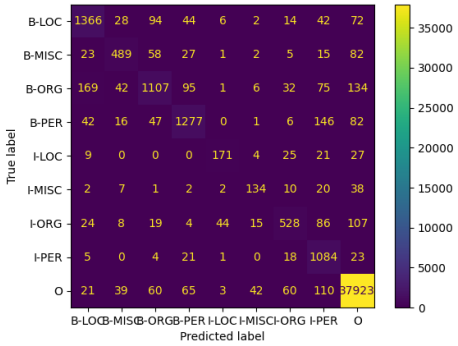

Figure 4: Confusion Matrix for the Logisitc Regression NER Classifier

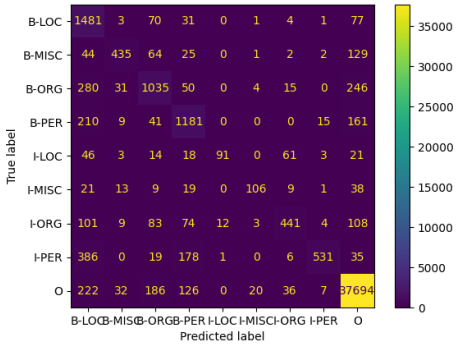

Figure 5: Confusion Matrix for the Naive Bayes NER Classifier

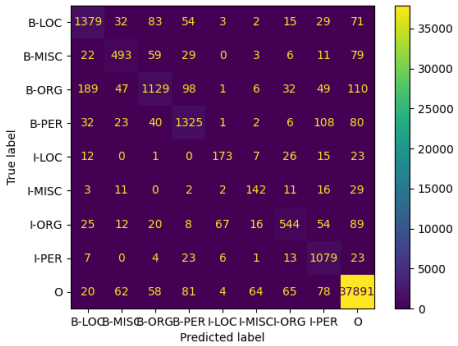

Figure 6: Confusion Matrix for the standard SVM NER Classifier

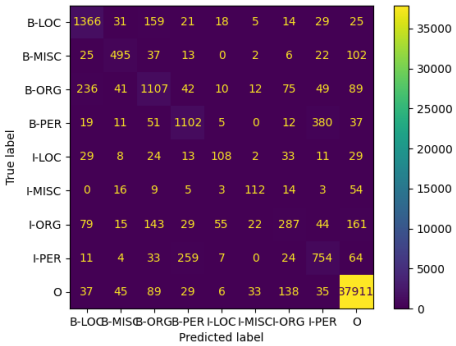

Figure 7: Confusion Matrix for the SVM with Embeddings NER Classifier

NERC-research preparation