Machine Learning for Named Entity Recognition and Classification Report

Susana Chen

Abstract

This project evaluates machine learning models for Named Entity Recognition and Classification (NERC) using the CoNLL 2003 dataset. Logistic Regression, SVM, and Naive Bayes models were trained with features such as tokens, embeddings, and context. Results show moderate performance, with Logistic Regression outperforming others in minority class recognition.

1 Introduction

This project aims to train and evaluate models for Named-Entity Recognition and Classification (NERC) using the CoNLL 2003 dataset (Tjong Kim Sang and De Meulder, 2003 , Section 1 and Section 2), which contains annotated Reuters news articles from 1996–1997. The focus is on classifying entities such as organizations, persons, and locations using the BIO tagging scheme.

A supervised machine learning approach was applied, using labeled data for model training. Results evaluated using macro-averaged metrics, show moderate performance for Logistic Regression and SVM models (macro f1: 0.66–0.74) and lower performance for Naive Bayes (macro f1: 0.42), attributed to poor recall. Logistic Regression, in particular, demonstrated strengths in precision and recall for minority classes like B-LOC and B-MISC, as seen in confusion matrices

2 Task and Data

2.1 Task

NER task description:

The Named Entity Recognition (NER) task involves recognizing and classifying entities in text into named entity categories such as persons, organizations, locations, miscellaneous entities, and non-entities. Using the CoNLL 2003 dataset, this task is presented with a supervised machine learning approach with BIO annotations. Each token in the dataset is labeled to demonstrate its named entity category.

A Logistic Regression model, a Naive Bayes model and a SVM model with hyper-parameter tunings were implemented, using one-hot features including tokens, part-of-speech tag, capitalization, frequency and n-gram features (considering both preceded and next-followed words). For the Logistic Regression model, word embeddings were also introduced for training.

Performance evaluation was implemented using standard metrics that include macro averages (precision, recall and f1 scores) by generating classification reports, as well as looking at visual representations in confusion matrices.

2.2 Dataset and Distribution

Data description and distribution

Within the data, each line represents a token with multiple annotation fields separated by whitespace, including the word itself, part-of-speech tag, chunk tag and named entity tag with position indication. The empty lines represent the boundaries of the sentences.

Categories of NER tags include LOC (Locations), ORG (Organizations), PER (Person names), MISC (names that are not in other categories).

Position indications of NER tags are

aligned with the BIO convention. Specifically, tag "I" stands for words inside of name entities; tag "O" for words outside of name entities; tag "B" for the first word of the second entity when two entities of a same type are next to each other.

The training and development datasets for the task demonstrate highly imbalanced distributions. The majority class "O", which refers to non-entity, dominates both datasets, constituting 83% of the total instances. Entity classes such as "B-ORG", "B-PER", "B-LOC", and "B-MISC" each accounts for approximately 2-3%, while classes like "I-LOC," "I-MISC," and "I-ORG" are least represented, only take up 0-1% of the data. This distribution poses challenges for the models in learning underrepresented entity classes, processes such as feature engineering could be considered in order to improve performance of the models.

2.3 Preprocessing

Preprocessing steps are completed in the codes, involving extraction of one-hot features and word embeddings, ensuring alignment of features with their respective NER labels. No extra adjusted files are created.

2.4 Evaluation Metrics

Given that the task involves multi-label classification, and that the predominant class 'O' is the least significant, macro averages were exclusively employed. This guarantees a relatively balanced evaluation across all classes, ensuring that the smaller, less frequent classes are not overlooked.

Evaluation was conducted using standard metrics, including macro precision, recall, and f1 score from classification reports, along with visual analyses of precision cases across different categories represented in confusion matrices.

3 Models and Features

3.1 Models

Model Descriptions:

Logistic Regression model is a common classification model that calculates the probability of a token to belong to a certain class. It does this by applying the sigmoid function to a sum of the input features. The sigmoid function can convert values into a range between 0 and 1, making it suitable for binary classification. In the case of Named Entity Recognition task with multiple classes, Logistic Regression is extended and generalized using the softmax function (Unpingco: 2022). In this project, the Logistic Regression model is trained on combinations of word embeddings with one-hot features for name entity classification.

Naive Bayes model is a classifier based on Bayes' theorem, mostly used in high dimensional text classification. The model assumes that features contributing to the results are conditionally independent of each other. Relatively few parameters are used in this model, resulting in a faster prediction speed. In this project, the Multinomial Naive Bayes classifier was used, as it is suitable for tasks where features that are text-based and discrete, such as token frequency and binary features like capitalization. Naive Bayes's assumption of independence is usually not sufficient in NERC tasks, which could lead to limited performance.

Support Vector Machines (SVM) is a model that maximizes the margin between different classes, in order to find a better decision boundary. It works by a hyperplane that separates classes to maximize the distance between the closest points of different classes, known as support vectors. In this way, the model variance is reduced. The regularization parameter C in SVM controls the balance between achieving a large margin and correctly classifying the training points. A larger C places more emphasis on the classification accuracy, while a smaller C emphasizes a wider margin (Unpingco: 2022).

For the SVM model in this project, SVC implementation was chosen since it is designed for binary and multi-class classification tasks, and that it is effective

in multi-dimensional spaces and managing sparse features. For the hyper-parameters tuning, a grid of C parameter distributions (0.1, 1, 10, 100) was set, and Grid Search was performed to evaluate each C parameter performance using 3-fold cross-validation with a fixed random state, so as to select the model with the best f1-score.

Features for the three models were vectorized by a DictVectorizer.

BERT is a pretrained transformer-based model that captures the bidirectional context of words in a sentence, making it highly effective for named entity recognition task. In this project, AutoModelForTokenClassification was used to initializes a BERT model tailored for token-level tasks.

The model's classification head was adjusted to match the number of ner tags. Random seeds are needed to ensure reproducibility of results across runs. Three seeds were tested to see how sensitive the model is to randomness in training.

3.2 Features

Six types of one-hot features and one type of dense features were considered and assessed.

The word itself is a critical feature for NERC since entities are often indicated directly by specific lexical information in the word itself (e.g. "EU").

The Boolean indicating whether the word starts with an uppercase letter was also included, designed as the capitalization feature. Since same entities, such as proper nouns, typically start with a capital letter (Hu et al.: 2024), this feature helps distinguish between general and entity-specific words (e.g. "apple", "Apple"). The feature is directly tied to the concerned token itself, since capitalization is a lexical property of individual words. In this way, capitalization is evaluated in the context of the specific token, so that the model is able to learn its importance related to the specific word and its surrounding context.

Token frequency in the dataset was considered as a numerical feature, helping the model recognize how common a token is in the dataset. Rare words are more likely to be named entities, as they often refer to unique persons, locations, or organizations. This feature provides indirect semantic information about the token's probability of being an entity.

Part-of-Speech (POS) tags were included, helping to identify the grammatical role of words, which is strongly associated with entity classes, since for example, proper nouns (NNP) are more likely to be entities. Moreover, certain verbs or adjectives can be information indicating name entities appearing in specific syntactical form.

Bigrams were introduced to enrich information of context by incorporating surrounding words. These features include the previous token bigram pair (e.g. "New_York" as the bigram_prev feature for the token "York") and the next token bigram pair (e.g. "York_is" as the bigram_next feature for the token "York"). Instead of treating bigrams as independent lexical units that excessively expand the feature space, they were used as concatenated token pairs to provide word dependency information without putting too much pressure on the size of the feature set. This feature design helps provide local context, which is essential for resolving ambiguities in token classification (Sung et al.: 2021). For example, in the case of the phrase "British lamb" (B-MISC, O), the model may rely on the token "lamb" to classify "British" as a miscellaneous entity. These features involve contextual dependencies that are not directly available from the CoNLL file but are derived from it.

Word embeddings are incorporated as features in the Logistic Regression model. They contain semantic and syntactic relationships between words in a dense vector format, encoding rich contextual information, such as similarity between words. They could also be useful in improving the model's performance

in classifying rare or unseen entities.

4 Experiments and Results

4.1 Feature Ablation

Detailed feature ablation was performed for the Logistic Regression Model.

First round of test:
At first, I tested the model with the baseline feature (the token itself), and combined it with one category of feature at a time (['token', 'is capitalized']), ['token', 'frequency'], ... , ['token', 'embeddings']).

Next, I inspected the results and noticed that the combination of token itself and the word embeddings produced the best f1-score (0.69) among all feature sets. Now a question occurred to me that, perhaps word embeddings encode semantic meaning of tokens more informative than raw token features. Adding raw tokens as a feature along with embeddings might introduce redundancy or even harm the model's performance.

Second round of test:
Therefore, I tested results using features sets ['token'], ['embeddings'], ['token', 'embeddings'], and the f1-scores were 0.49, 0.67, 0.69 respectively. It turned out that the combination of raw token features and word embeddings did contribute to better performance.

Third round of test:
On the basis of the feature set ['token', 'embeddings'], I introduced three-feature sets with one extra feature for each (e.g. ['token', 'embeddings', 'pos']).

Based on the results so far from the completed tests, feature sets with 'frequency' produced the worst f1-scores, and that the output from ['token', 'embeddings', 'frequency'] (f1-score: 0.6922) was worse than that of ['token', 'embeddings'] (f1-score: 0.6934). Therefore, I decided to remove the feature 'frequency'.

In this particular round of test, the feature set with the highest f1-score (0.72 for ['token', 'embeddings', 'bigram_prev']) was preserved. To continue with the test, four-feature sets were formed (e.g. ['token', 'embeddings', 'bigram_prev', 'pos']).

The rest of the test rounds were in the same pattern.

With all combinations tested, the feature set ['token', 'embeddings', 'bigram_prev', 'bigram_next', 'is_capitalized', 'pos'] exhibited the best performance with the highest f1-score (0.74).

Analysis on Feature Effects:
Results from the ablation testing process indicated that raw token features are essential for the task but needs complementary features to improve results. Features including capitalization, pos, bigrams with the preceding token, word embeddings are beneficial for the model's performance (e.g. feature 'bigram_prev' improved the f1-score by 0.0135 over token alone, 0.0315 over token + word embeddings).

The feature 'bigram_next' is a moderate one, which harmed the performance when being combined with 'token' only (f1-score decreased by 0.0065 comparing to 'token' alone), but contributed moderately positive impact when combined with embeddings (improved the f1-score by 0.0199 over token + word embeddings).

The feature 'frequency' is a harmful one, given that its combination with 'token' decreased the f1-score by 0.0184 compared to 'token' alone, and that its combination with ['token', 'embeddings'] produced worse results (f1-score 0.0012 lower).

What's noticeable from the interactions of features is that, embeddings alone are highly effective but benefit significantly from the addition of contextual (bigram_prev, bigram_next) and structural (is_capitalized) features.

Ablation tests were also carried out for the Naive Bayes model and the SVM model,

results show that ['token', 'bigram_prev'] is the most suitable feature set for the former model, and ['token', 'is_capitalized', 'bigram_prev'] for the latter.

4.2 Evaluation

Three models were evaluated (Logistic Regression Model, Naive Bayes Model, SVM Model), and due to constant memory errors, same size of data subsets (50,000 for training; 30,000 for testing) were used in the evaluation. Features selected for each model were based on the feature ablation results. Hyper-parameter tuning for the SVM Model was performed, results will be compared between the untuned model and the tuned one.

Logistic Regression Model (Features: ['token', 'embeddings', 'is_capitalized', 'bigram_prev', 'bigram_next', 'pos']):

	precision	recall	f1-score	support
B-LOC	0.87	0.85	0.86	1073
B-MISC	0.82	0.80	0.81	539
B-ORG	0.73	0.72	0.73	783
B-PER	0.83	0.74	0.78	1076
I-LOC	0.69	0.62	0.65	150
I-MISC	0.79	0.53	0.63	202
I-ORG	0.65	0.47	0.55	439
I-PER	0.61	0.76	0.68	763
O	0.99	1.00	0.99	24975
accuracy			0.95	30000
macro avg	0.77	0.72	0.74	30000
weighted avg	0.95	0.95	0.95	30000

Figure 1: Logistic Regression Classification Report

For macro averages, the precision is 0.77, and the recall is 0.72, resulting in an f1-score of 0.74. The confusion matrix indicates strong performance for the majority class "O", and demonstrates moderate performance for some minority classes like "B-LOC" and "B-MISC", where both precision and recall are above 0.80, showing its ability to categorize these classes reasonably well. Similarly, "B-PER" and "B-ORG" show comparable performance, despite of some misclassifications into other categories, for example, 20% of "B-PER" instances are classified as "I-PER".

However, performance significantly deteriorates for minority classes with sparse representations, such as "I-MISC," "I-LOC," and "I-ORG". These classes have lower recalls, indicating that the model struggles to correctly identify these entities. For example, "I-MISC" achieves only 0.63 for recall, and "I-ORG" struggles with an f1-score of 0.55 due to frequent false negatives and misclassifications into "O" or other similar tags.

Naive Bayes Model (Feature set: ['token', 'bigram_prev']):

	precision	recall	f1-score	support
B-LOC	0.90	0.55	0.68	1073
B-MISC	0.89	0.19	0.31	539
B-ORG	0.86	0.17	0.29	783
B-PER	0.99	0.19	0.32	1076
I-LOC	0.75	0.42	0.54	150
I-MISC	0.97	0.15	0.26	202
I-ORG	0.93	0.17	0.29	439
I-PER	0.96	0.11	0.19	763
O	0.87	1.00	0.93	24975
accuracy			0.88	30000
macro avg	0.90	0.33	0.42	30000
weighted avg	0.88	0.88	0.84	30000

Figure 2: Naive Bayes Classification Report

The macro averages for the Naive Bayes model show a significant performance imbalance, with precision at 0.90, recall at 0.33, and an f1-score of 0.42. While the model demonstrates high precision (0.75 0.99) for most of the classes, its recall is notably low, leading to a relatively low f1-scores (0.19 0.54) for most minority classes.

The confusion matrix shows that majority of the named entities are misclassified as class "O", causing prevalence of false negative cases and low recalls. Among those classes, "B-LOC" performs relatively well with an f1-score of 0.68, though 44% of the instances are identified as "O". For classes like "B-MISC" and "B-ORG", recall is notably low, as the model frequently predicts majority of these instances as "O" and minority of them as other classes. "B-PER" exhibits high precision (0.99) due to minimum cases of false positive, but suffers from an extremely low recall (0.19), resulting in substantial misclassifications.

Similar results are shown for intermediate classes ("I-*"), reflecting the model's strong bias toward the majority class. The majority class "O" dominates predictions with perfect recall (1.00) and an F1-score of 0.93.

This overwhelming bias towards "O" severely undermines the model's ability to recognize minority classes effectively.

SVM Model (Feature set: ['token',

'is_capitalized', 'bigram_prev']):
Results Without Hyper-Parameter Tuning:
The SVM model without hyper-parameter tuning achieved a macro f1-score of 0.6358. While the model performed well for the majority class "O" (f1-score = 0.99), its performance on minority classes such as "I-ORG" (f1-score = 0.52) and "I-PER" (f1-score = 0.35) was poor. Notably, the precision for "B-PER" was extremely low (0.37), indicating significant misclassification issues for this category, despite a very high recall (0.98).

Results With Hyper-Parameter Tuning:

	precision	recall	f1-score	support
B-LOC	0.89	0.75	0.81	1073
B-MISC	0.87	0.58	0.70	539
B-ORG	0.78	0.56	0.65	783
B-PER	0.38	0.94	0.54	1076
I-LOC	0.77	0.64	0.70	150
I-MISC	0.90	0.51	0.65	202
I-ORG	0.92	0.41	0.57	439
I-PER	0.74	0.25	0.37	763
O	0.99	0.99	0.99	24975
accuracy			0.93	30000
macro avg	0.81	0.63	0.66	30000
weighted avg	0.95	0.93	0.93	30000

Figure 3: Tuned SVM Classification Report

After tuning the hyper-parameter C using grid search, the macro f1-score improved to 0.6647, demonstrating better generalization across classes. Improvements in f1-scores are recognized for several minority classes, such as "I-ORG" (f1-score = 0.57 compared to 0.52) and "I-PER" (f1-score = 0.37 compared to 0.35). From the confusion matrix, we can also see that the amount of true positive cases for "I-*" classes increases.

Both versions of results show severe false positive cases for the class "B-PER".

BERT Model:

Metric	Seed 0	Seed 42	Seed 123
Overall Precision	0.937000	0.939500	0.930000
Overall Recall	0.946800	0.944500	0.943300
Overall F1-score	0.941800	0.939800	0.936600
LOC F1-score	0.962900	0.966500	0.960800
MISC F1-score	0.842700	0.835200	0.830300
ORG F1-score	0.915200	0.913200	0.912100
PER F1-score	0.982100	0.976500	0.970400

Figure 4: BERT Results

Discussion on Variations:
Seed = 0 achieved the best overall f1-score (0.9418), while Seed = 123 had the lowest (0.9366). The variation in macro f1-score is minor, showing consistent predictions.
LOC and PER classes consistently present high f1 scores (0.9608 0.9821), while MISC shows the lowest f1 (0.8303 0.8427), indicating it might be more challenging to classify accurately, potentially due to fewer training samples.

The choice of random seed slightly impacts evaluation metrics, especially for challenging or minority classes like MISC. The overall performance remains strong across seeds, indicating stability in BERT fine-tuning.

5 Error Analysis

The confusion matrix of the Logistic Regression model reveals specific challenges in distinguishing between certain entity types. Positions of entities are frequently confused, for example, the "B-PER" class demonstrates relatively strong performance but is still misclassified as "I-PER" in 228 instances, likely due to weak boundary detection between beginning and inside labels. Moreover, location entities and organization entities are also easy to confuse. For example, the "B-LOC" class is frequently misclassified as "B-ORG" (80 instances).

The randomly chosen sample (100 examples) correctly classified 71% of the instances, excluding the dominant 'O' class, leaving 29% misclassified.

Among the misclassified examples, there is a notable pattern. Correct identifications of entity class paired with incorrectly detected positions are prevalent, with a proportion of 45%, probably the result of lack of dependency information. For instance, "Africa" was predicted as "I-LOC" instead of "I-ORG", while the bigram_next feature includes the next token "Ltd", a clear indication of organization-related entity. Such error could be the result of insufficient interpretation of the contextual information from the one-hot features. Moreover, 69% cases of such pattern are in class "PER" (e.g.

"Tolunay" was misclassified as "I-PER" rather than "B-PER"), showing that the challenge lies mainly within identifying positional boundaries for personal names.

Among the correctly classified examples, it is worth noticing that 85% of the case are in class "LOC" and "PER". For example, "Britain" and "Chile" were correctly identified as "B-LOC", suggesting that the model identifies certain classification patterns for well-represented locations. The token "Glenn" was also accurately classified as "B-PER", indicating that the model has sensible reliability in classifying some of the personal names.

Additionally, the confusion between organization-related labels ("B-ORG", "B-LOC") and others like "B-MISC" suggests a need for more contextual features, such as richer dependency parsing.

6 Discussion

Imbalanced performance with significant bias towards the majority class "O" from the Naive Bayes model is noticeable, which requires further implementation of feature engineering, such as incorporating word embeddings or positional features.
Due to the constant occurrence of memory errors, I had to use 1/4 subset of the training data and 60% of the testing data to ensure that the kernel does not break. I have tried to run the full dataset during the model training in different computational environments including Kaggle and Google colab, none of them work eventually.

7 Conclusion

Logistic regression proved to be the most effective for NERC in this project, achieving robust results with carefully selected features. Challenges included imbalanced data and sparse minority classes. Future improvements should involve preprocessing to balance data distributions, as well as exploring advanced embeddings and contextual features to further improve classification performance.

References

Chul Sung, Vaibhava Goel, Etienne Marcheret, Steven Rennie, and David Nahamoo. 2021. CNNBiF: CNN-based Bigram Features for Named Entity Recognition. In Findings of the Association for Computational Linguistics: EMNLP 2021, pages 1016–1021, Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

Unpingco, J. (2022). Python for probability, statistics, and machine learning (Third edition). Springer. https://doi.org/10.1007/978-3-031-04648-3

Yan Hu, Qingyu Chen, Jingcheng Du, Xueqing Peng, Vipina Kuttichi Keloth, Xu Zuo, Yujia Zhou, Zehan Li, Xiaoqian Jiang, Zhiyong Lu, Kirk Roberts, Hua Xu, Improving large language models for clinical named entity recognition via prompt engineering, Journal of the American Medical Informatics Association, Volume 31, Issue 9, September 2024, pages 1812–1820.

Theoretical Component

Definition of Machine Learning:
Machine learning is a process of training artificial intelligence on existing data to imitate human's learning process. Specifically, machine learning involves making decisions on prediction or classification, and with labeled and unlabeled data, the process can be classified as supervised or unsupervised.

Feature Selection:
The selected features include the token itself, which is critical for the NERC task since entities are often directly indicated by specific words. Capitalization, which distinguishes proper nouns from general words, is another key feature, considering that named entities frequently start with uppercase letters.

Representation of advanced features includes Word embeddings, which provide semantic and syntactic relationships between words in a dense vector format, capturing rich contextual information. This is especially valuable for managing unseen or rare entities. Bigrams are also advanced features that identify the relationships between a token and its immediate context (preceding and following words), minimizing ambiguities in entity classification.

Motivations and Explanations:
The inclusion of basic features such as tokens and capitalization is driven by their simplicity and effectiveness in identifying proper nouns and specific named entities. Advanced features like word embeddings and bigrams were selected for their ability to encode richer contextual dependencies. For example, embeddings improve model's ability in identifying dependency information, while bigrams consist of local context crucial for distinguishing similar tokens in different entity roles and positions. The ablation results of the Logistic Regression model demonstrated that combining token features with embeddings and bigrams yielded the best performance (macro f1-score: 0.74), validating the motivations behind their inclusion.

Assignment 2

1.A.1 The goal of Subtask B was to classify the sentiment of a message, the message can be a tweet or a SMS. The task was designed to categorize messages as positive, negative, or neutral. When messages conveyed both positive and negative sentiments, the dominant one was chosen as the correct label. The system aimed to handle the challenges of processing informal text by analyzing and classifying the overall sentiment. The input of the system was the particular message (a tweet or a SMS), the desired output is the classification label (positive, negative, neutral).

1.A.2 Useful features:
Token itself, choice of words (e.g. "nice" "Nice"): Identifying specific words and their implications;
POS tag: Understanding the role each word plays in a sentence can help understand sentiment-bearing terms;
Context of the token (e.g. "very good" "not good"): Considering surrounding words;
Chunk tags (e.g. [very good], [not good]): Phrases can express sentiment collectively;
Punctuation ("!" indicates aggressiveness): Usage of exclamation marks or other punctuation can indicate sentiment intensity;
Shape of tokens (all capitalized): Capitalization or all caps can show emphasis;
Sentence length: shorter sentence could be more emotional.

1.A.3 These features can be presented to the system by encoding each token with relevant feature vectors (e.g. token position, POS tag, surrounding context). For example, a tweet could be represented using a set of vectors using word n-grams, shape of tokens, chunk tags.

1.B.1 Coreference resolution is the task of identifying and grouping references of the same entity.

Input: A document containing references (e.g. noun phrases, pronouns, named entities) that could connect to entities.
Output: A set of clusters, where each cluster contains all references of the same entity.

1.B.2 Exact String Match:
1. Definition: Links references if they have identical forms.
Why it's useful: Ensures high precision.
Example: The mention "The Eiffel Tower" in two

different sentences can be linked.

2. Definition: Links mentions based on shared head words, in the case that their modifiers do not conflict.
Why it's useful: Improves flexibility in linking mentions with slight differences.
Example: Links "The university" and "Harvard University," given that there is no conflicting information.

3. Definition: Links pronouns and names (he, she, it) by ensuring agreement in attributes like gender and number.
Why it's useful: Could be helpful when dealing with ambiguous pronouns by referring to context.
Example: Links "John" to "he" based on gender and word position in the sentences.

1.B.3
Feature 1: Exact String Match
The system scans the text for mentions with identical forms. It directly links mentions without additional processing.

Challenges: Cannot handle paraphrases and variations like "The president" vs. "President Obama."

Feature 2: Pronoun Agreement
The system extracts attributes (gender, number, animacy) from the entities and compares them to that of pronouns. Unlike humans, the system relies on explicit rules for attributes.

Challenges:
Ambiguity in the connections of pronouns when multiple references are nearby. For example, there could be a case where "he" is in a sentence with more than one males.

Assignment 3

1. How a KNN Classifier Works:
The K-Nearest Neighbor (KNN) classifier assigns a new instance to a class based on the majority class of its k closest neighbors in the feature space. The extent of the distance to its neighbors is typically determined using a distance metric that includes the Euclidean distance. KNN is a lazy learning algorithm, which means it does not build a model during training, but instead classifies directly using the training data.

2. Explanation of classification with different K values
When k=1, the instance will be classified as a green triangle, since the closest neighbor is a green triangle.

When k=3, the new instance will probably be classified as a green triangle if most of the three closest neighbors (inside the red circle) are green triangles.

When k=5, the classification depends on the majority class among the five nearest neighbors. If three green triangles and two yellow squares are the nearest, the new instance will still be classified as a green triangle.

3. Effect of different K values
When k=1, the model is very sensitive to noise, since the classification depends on a single instance, which can lead to overfitting.
When k is very large (e.g. equal to the total size of the data set), the classifier essentially sees all data as average, ignoring local patterns and resulting in underfitting. The model could fail in identifying important distinctions between classes.

A Example Appendix

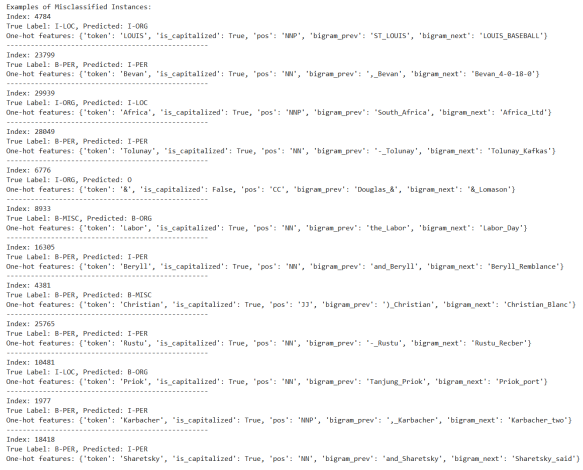

Figure 5: Samples of Misclassified Cases

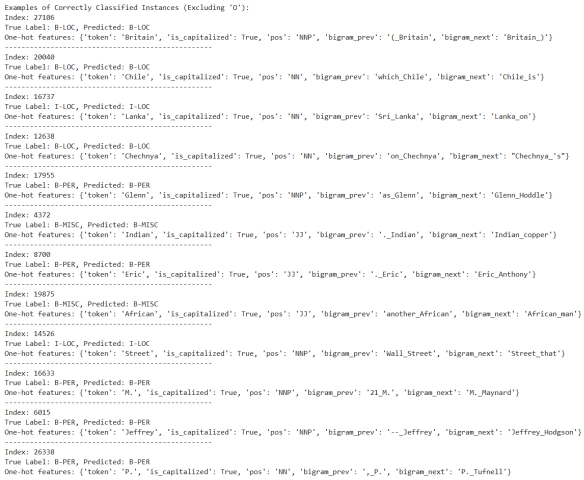

Figure 6: Samples of Correctly Classified Cases

NERC-research preparation

Evaluation Metrics: Precision, recall, and f1-score are used to assess model performance. Precision answers the question of all entities predicted as a certain label, how many were correct; recall answers the question of all actual entities in the dataset, how many were correctly identified. F1-score is used to measure the balance between precision and recall.

High precision scenario: Email spam detection. In this case, you want to minimize false positives because you don't want to check the spam regularly and don't want to miss important emails because of them likely to be detected as spams.

High recall scenario: When detecting harmful or threatening messages on a social media platform, high recall is crucial to capture as many relevant threats as possible, even if it leads to a few false positives.

Confusion matrix analysis was conducted to observe where errors commonly occur (e.g. confusion between 'B-ORG' and 'I-ORG').

Token-level evaluation measures each token individually, which might miss entity-level context but provides fine-grained analysis, while span-level evaluation evaluates complete entity spans (e.g. the noun phrase "David Bowie" as B-PER I-PER) rather than individual tokens. This can be more helpful in real-world applications but is more challenging, since more specific handling is needed.

When only parts of a BIO entity tag could be identified correctly (e.g. correct class but wrong position), solutions can involve defining how to score degree of matching or weighting different types of error.