Machine Learning for Named Entity Recognition and Classification Report

Dorielle Lonke

Abstract

The task of named entity recognition and classification has been thuroughly investigated in terms of effective learning models as well as relevant features. This report describes the results of implementing and evaluating different models trained for named entity recognition and classification, using a variety linguistic features represented as one-hot vectors as well as word embeddings. The systems are trained on selected features and evaluated in terms of precision, recall and harmonic mean (f1-score).

1 Introduction

The task of Named Entity Recognition and Classification (NERC) concerns the identification of named entities, i.e. names of people, places, or organizations that appear within a text. This experiment makes use of CONLL2003 dataset, taken from the CoNLL 2003 shared task which focused on language-independent named entity recognition in English and German. (Sang and De Meulder, 2003). I experiment with training four models - Linear Regression, Naive Bayes, Support Vector Machines (SVM) and Conditional Random Fields (CRF), and compare their overall performances. Prior to the training process, I identified and extracted a selection of linguistic features from the CONLL dataset which were employed in different settings and combinations alongside pretrained word embeddings. The experiment shows that the SVM-based NER system, trained with combined one-hot representations and pretrained word embeddings, yielded the best results.

2 Related work

Since first investigated in the 90s by Grishman and Sundheim [1], the task of named entity recognition has been organized for a variety of languages, including under-resourced languages such as Bengali (Ekbal and Bandyopadhyay, 2008) and Lithuanian (Kapočiūtė-Dzikienė et al., 2013). As described in a recent survey on NER advances (Yadav and Bethard, 2018), many of the early NER systems were based on handcrafted rules and utilized straightforward orthographic features, and employed supervised machine learning models such as SVMs, CRFs and decision trees. These approaches are still widely used; Kapočiūtė-Dzikienė et al. make use of orthographic features pertaining to the particular morphology of Lithuanian names, combined with syntactic information and external gazetteers, which are represented in a vector taken together with the token and label. Ekbal and Bandyopadhyay combine common Bengali name prefixes as features with contextual embeddings in an SVM-based NER system. In their survey, Yadav and Bethard note that recent developments in NER tasks introduce unsupervised methods such as Neural Networks with the potential of learning latent features. Recent work on the interface of NER and language models examines how the task of named entity recognition can be further expanded with new advancements in LLMs (Katz et al., 2023). The authors' objective is to expand the capacities of NER systems in fine-grained, zero-shot entity recognition and retrieval tasks even further.

3 Task and Data

3.1 Task

The task at hand is a Named Entity Recognition and Classification (NERC) task, which is carried out by training several machine learning models on the CONLL2003 dataset, with the ultimate goal of identifying the best model for the task by comparing the evaluation results. The CoNLL data is represented in tsv format whereby each line represents a word and each column represents a feature of the word. It contains four columns, corresponding to the word, its part of speech, its chunk infor-

[1] https://www-nlpir.nist.gov/related_projects/muc/

mation, and its named entity tag + BIO tag. An empty line represents a sentence boundary. The categories used in the CONLL2003 dataset are ORG (organization, e.g. *Disney*), LOC (location, e.g. *Italy*), PER (person, e.g. *Clinton*), and MISC (miscellaneous, i.e. other named entity that does not fit into the description, such as origin adjectives or events, i.e. *Italian* or *English Country Championship*). The annotations are named entity tags which mostly follow the MUC-6 conventions (Grishman and Sundheim, 1996), combined with the IOB (BIO) tagging scheme (Ramshaw and Marcus, 1999), which indicates whether a word of a certain type is inside (I) a named entity, the beginning (B) of a named entity, or outside (O) of a named entity. O usually indicates that the word is not a named entity at all.

3.2 Dataset and Distribution

The distribution of the NER labels in the CoNNL2003 training set is as follows: the largest category is O, i.e. no entity, with 42,120 instances. This is technically a non-category as it does not represent a named entity. However, these represent a significant portion of natural language and as such used in the training process. The largest named entity class is B-PER, with 1842 instances, followed closely by B-LOC with 1837 instances. The next classes are B-ORG with 1341 instances, I-PER with 1303 instances, B-MISC with 922 instances, I-ORG with 751 instances, I-MISC with 346 instances and finally I-LOC with 257 instances. The distribution follows a pattern wherein most tagged entities are identified as the beginning of a named entity, person being the most prominent category both for B and I. MISC is the least prominent out of the four, ranked just below I-PER. The best represented class is B-PER (and PER in general). The class with the least amount of data is generally MISC, but there are the least instances of I-LOC - possibly due to the fact that most locations are identified as single words (see A1 for a graphic representation).

3.3 Preprocessing

The data was minimally preprocessed, only featuring a structural change to the CONLL files to include the features. In terms of the tokens, I decided not to remove any tokens from the dataset, keeping special characters such as punctuation mark and parentheses, as these might occur as part of named entities and could have effect on the learning process. The CONLL files were prepared for the task by applying a preprocessing script that extracts the features and adds them as columns to the files, alongside a header containing titles for the original data (*Token*, *POS*, *Chunk* and *Gold*) and feature names, which was used for extracting specific sets of features during the training. The preprocessing script was applied once in the beginning of the experiment, and the updated files were used from that point onward.

3.4 Evaluation Metrics

I evaluated the system in terms of its precision, recall and f-score (harmonic mean). The precision score indicates how many of the cases returned by the system are relevant - this is calculated by looking at the number of true positives out of the total number of positives - i.e. the number of cases returned by the system. Recall indicates how many of the relevant cases were returned by the system - this is calculated by looking at the number of true positives out of the sum of true positives and false negatives. The f-score, or harmonic mean, is calculated by looking at twice the product of the precision and the recall scores, divided by the sum of the precision and the recall scores. The harmonic mean gives an overall picture of the performance of a system - the higher the precision and recall scores are, the higher the harmonic mean. I am calculating the overall precision, recall and f1-scores by taking their macro-averages, in order to mitigate the data imbalance caused by a disproportionate representation of the majority class O.

The evaluation was done at the token level - i.e. calculating precision, recall and f-score per category by looking at the system's performance for each individual token, . For example, if a named entity consists of 'Amsterdam (B-ORG) Global (I-ORG) Health (I-ORG) Institute (I-ORG)', but the system labelled 'Institute' as B-ORG, 'Global', and 'Health' as I-ORG, and Amsterdam as B-LOC, this would be considered as though the system was correct for 'Global', and 'Health', but was wrong for 'Amsterdam' and 'Institute'. I decided to keep the BIO tags in the evaluation and look for exact matches, as my experiment featured a comparison of different models and focused on potential differences in the overall treatment of named entities. It should be noted that if the goal is to find all mentions of some named entity, an evaluation at the span level, or one that allows for partial matches,

might be preferable.

4 Models and Features

4.1 Models

In this experiment I employed four different machine learning models, evaluating them independently on different configurations of the feature set and comparing the results. The baseline model used in this experiment is the Logistic Regression (LR) model, with an initial set of four baseline features: *token*, *pos*, *allcaps* and *cap_after_lower* (explained in more detail in §4.2). Logistic Regression is a probabilistic model which works well with discrete classes such as the NER categories. Additionally, I trained a Naive Bayes (NB) classifier and a LinearSVC (SVM), two commonly used classifiers. The Naive Bayes classifier is also a probabilistic model which selects, given the observed features, the most likely class to have generated these features, using an assumption of conditional independence of the features in the feature set. SVM classifies the data by creating decision boundaries with maximal margins in high-dimensional spaces. Finally, I trained a more advanced CRF model, which also predicts the most likely class given the observations, but takes into account dependencies between neighboring components, making it effective for tasks such as NER in which tokens originate in sequential data. According to the survey performed by Yadav and Bethard, SVMs and CRFs are commonly used models for NER tasks, as well as Hidden Markov Models (HMM), statistical models which, like SVMs, makes use of local information. In terms of its behavior, LR is closely related to maximum entropy models which are common practice in NER tasks, making it a suitable and effective model for NER (Jiang and Zhai, 2006).

4.2 Features

I defined eight different features extracted directly from the CONLL data, based mostly on orthographic properties of the token, its preceding token, or its following token. Firstly, I made use of the syntactic features provided in the original dataset. I defined three features as the ***token*** itself, together with its ***part-of-speech*** and its syntactic ***chunk***. Additionally, I made use of five orthographic features, for which I tested different combinations. By inspecting both content and form of the most frequent words per category, I was able to obtain an initial idea of the possible orthographic features. For one, proper names (i.e. people, places or organizations) are usually capitalized, and organizations are often written as acronyms or initials which are all uppercase. I defined two features related to word case: ***allcaps*** which is a binary feature determining if a word is upper- or lower-case, and ***cap_after_lower*** - a binary feature which indicates whether a word is capitalized when coming after a fully lowercase word, i.e. mid-sentence. I included this constraint to ensure that this feature excludes words that are at the start of a sentence. Further, I noticed that the instances the MISC category, which captured many origin adjectives such as *Canadian*, *Scottish*, *Mexican*, etc., follow a common pattern of ending with *ish*, *ian*, or *an* - English suffixes indicating the sense of 'belonging to' or 'originating from'. I thus defined the binary feature ***demonym*** which indicates if a token contains such a suffix, as this sense of having an origin correlates with persons or entities. Inspired by the *Corporate-Suffix* feature (Chieu and Ng, 2002), I defined the binary feature ***comp_suf*** which determines whether a token is followed by one of the company markers 'Ltd.', 'Inc.', 'LLC', 'Co.' or 'Corp.' - indicating a named entity belonging to ORG. Finally, I included the binary feature ***poss_mark***, which indicates if a token is followed by the English possessive marker *'s*. The possessive marker is suffixed to the possessive form - a noun or noun phrase, and usually signifies agency, or at least some capacity of possession which is reserved for people or organizations. In addition to this set of features, which were represented as one-hot vectors, I made use of word embeddings taken from the word2vec pre-trained Google News corpus [2]. Compared to the sparse and high-dimensional one-hot representations, word embeddings are low-dimensional, dense vector representations of words that capture the contextual information of a word. For the task at hand, such contextual information is useful for determining whether a word that might orthographically appear like a named entity is actually a named entity by considering its surrounding words.

5 Experiments and Results

In this section I evaluate the baseline model - Logistic Regression trained with four features - *token*, *pos*, *allcaps* and *cap_after_lower* - compared to the results of training also Naive Bayes and LinearSVC classifiers with an extended set of features, includ-

[2] https://www.kaggle.com/datasets/adarshsng/googlenewsvectors

ing word embeddings as features, SVM with ablated features and finetuned hyperparamaters, and the results of training a CRF.

5.1 Evaluation

The baseline model had a precision score of 70% and 65% recall. It did relatively well compared to the other systems trained with one-hot features, and even outperformed the SVM trained with word embeddings as features, which only had a precision score of 67% and 62% recall. This might be related to the fact that the initial set of features already included significant syntactic and orthographic information that contributed to the recognition of the named entities. When using the baseline feature set with SVM, there was only slight improvement - the same precision score of 70% was noted, with a 67% recall score and 68% f1-score.

Class	precision	recall	f1-score
B-LOC	0.839	0.733	0.782
B-MISC	0.787	0.612	0.689
B-ORG	0.640	0.568	0.602
B-PER	0.683	0.562	0.616
I-LOC	0.656	0.482	0.556
I-MISC	0.630	0.569	0.598
I-ORG	0.696	0.489	0.572
I-PER	0.442	0.878	0.588
O	0.982	0.983	0.983
macro-average	0.706	0.653	0.665

Table 1: Classification results for the baseline LR model.

When including more features in the feature selection, an improvement was noted for LR and SVM. Table 2 shows the macro-averages for the precision, recall and f1-scores of the LR, NB and SVM models. The features used in this iterations were all eight features mentioned in §4.2. All three models have a comparable precision score, with 73% for LR, 74% for NB and 72% for SVM. In terms of recall, the Naive Bayes classifier performed significantly worse than its counterparts. This might be due to the fact unlike LR and SVM, which are discriminative learning models, NB is a generative learning model. In that sense, it is good with ensuring that out of all returned cases, most were correctly returned, but worse with identifying all the relevant cases for a given class. Another type of features I experimented with was word embeddings. I conducted two experiments with the SVM model. The first one included training the SVM with only word embeddings as features. Compared

Model	precision	recall	f1-score
LR	0.733	0.675	0.691
NB	0.747	0.512	0.563
SVM	0.721	0.703	0.706

Table 2: Overall macro-averages for LR, SVM, and NB with the extended feature set.

to the baseline model, the result demonstrates a slight degradation in all metrics - with a precision score of 67% (compared to 70% in the baseline model), a recall score of 62% (compared to 65%) and a f1-score of 64% (compared to 66%). This is not a stark difference that could indicate that the sparse, high-dimensional one-hot representations of the syntactical and case-related features are as effective as the low-dimensional, dense embeddings for this task. I then experimented with combining the one-hot features with the word embeddings. In this iteration, I included the baseline features (*token*, *pos*, *allcaps*, *cap_after_lower*), as well as word embeddings. The results of this experiment showed remarkable improvement on all metrics. Table 3 below shows the results of the combined feature SVM-based system for each class. As a final experiment, I trained a CRF model on the same training data. The CRF takes dependencies into consideration, and was able to learn the tokens in their surrounding context. The CRF demonstrated an improvement on all metrics compared to the baseline model, especially in the recall score - with 74% recall compared to 65% in the baseline model. However, it did not perform as well as the combined feature SVM-based system.

Class	precision	recall	f1-score
B-LOC	0.900	0.939	0.919
B-MISC	0.943	0.944	0.943
B-ORG	0.912	0.885	0.898
B-PER	0.941	0.931	0.936
I-LOC	0.820	0.836	0.828
I-MISC	0.937	0.768	0.844
I-ORG	0.881	0.792	0.834
I-PER	0.903	0.923	0.913
O	0.996	0.998	0.997
macro-average	0.915	0.891	0.901

Table 3: Classification results for the SVM model trained with combined one-hot features and word embeddings.

Model	precision	recall	f1-score
Baseline LR	0.706	0.653	0.665
Combined SVM	0.915	0.891	0.901
CRF	0.772	0.746	0.765

Table 4: Overall macro-averages for the baseline LR, the combined feature SVM, and the CRF models.

5.2 Feature Ablation

In this process I examined the effect that the different features had on the SVM model. I was interested in checking the contributions of the orthographic features (*demonym*, *comp_suf* and *poss_mark*) to the overall performance of the model. I started by training an SVM using only the *token*, and then using the baseline features (*token*, *pos*, *allcaps* and *cap_after_lower*). Then, I introduced feature sets consisting of the token + one of the orthographic features. Then I tried different combinations of the orthographic features with the case-related features (*allcaps* and *cap_after_lower*), and finally included a feature set with the syntactic features *pos* and *chunk* alongside the case-related features, effectively expanding the baseline features to include additional syntactic information. The results showed an interesting pattern: the feature combinations containing only orthographic features and no syntactic features demonstrated an increase in the precision score compared to the baseline features but a significant decrease of almost 10% in the recall score, and a subsequent decrease in the f1-score. The best feature combination turned out to be one which incorporates both orthographic and syntactic features - either a feature set consisting of the token, syntactic information and case-related features, or the entire extended set of all eight features - which had nearly identical results.

5.3 Hyperparameter Tuning

In order to examine the settings of my SVM model, I conducted hyperparameter tuning and trained a hypertuned LinearSVC model with the complete feature set. This was done with the *HalvingRandomSearch* method of *scikit-learn* [3], a search strategy that iteratively selects the best candidates for the parameters of a model. By initializing the search method with distributions for the regularization parameter C, the loss function and the tolerance for stopping criteria, I extracted the best parameters and then passed those to a new LinearSVC instance and trained it on the data using all features. The results showed an insignificant decrease of 2% for the precision score, compared to the default SVM, and no changes to the recall and f1-score. In light of these results, I decided to keep the default settings for the SVM model when later training it using combined sparse and dense features.

[3] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.HalvingRandomSearchCV.html

Features	precision	recall	f1-score
Baseline	0.70	0.67	0.68
Token+orth.	0.73	0.57	0.63
Token+cap+orth.	0.73	0.60	0.65
Token+cap+syntax	0.71	0.70	0.70
All features	0.72	0.70	0.70

Table 5: Overall macro-averages for SVM with baseline features, SVM with a combination of token and one or more orthographic features, combined orthographic and case-related features, and SVM with combined syntactic and case-related features as well as the full feature set.

6 Error Analysis

The error analysis was carried out for two systems: the first is the SVM-based NER system trained with the baseline feature set, and the second is the SVM-based NER system trained with combined one-hot representations and word embeddings. I was curious to compare the results for both correct and incorrect results for the two systems, and attempt to witness traces of the effect of the word embeddings that caused such significant improvements in all evaluation metrics. As mentioned before, I included the majority class O but I evaluated using the macro-averages for the precision, recall and f1-scores in order to mitigate its effects. This category was consequently prominent in the confusion matrix, but I will not be addressing any errors which consist of mislabelling one of the classes PER, ORG, LOC or MISC as O in the following analysis.

In the case of the first system, the classification report shows that the system had relatively low accuracy scores for the classes containing an I tag - I-PER, I-MISC and I-LOC scored lowest in terms of precision. In terms of recall, however, the highest score belonged to the I-PER class. This indicates that the system was mostly able to select this class

for the relevant cases, but it also had a high amount of false positives for that class. Examples of false positives occur prominently in a confusion between B-PER and I-PER, where the system labels a token which is tagged as B-PER as I-PER. This preference is even evident in confusions between the category O and I-PER. This could be due to latent patterns learned by the system which were not learned when more elaborate orthographic features were included. When comparing this phenomenon with the second system, indeed this confusion was significantly reduced, possibly by means of introducing dense features which introduced new information that the system was able to learn and associate with the PER class. Nevertheless, there was still some confusion between B-PER and I-PER in the second system, wherein about 100 cases out of 1617 in the B-PER class were labelled as I-PER by the system. The system displayed many recurring labelling errors with respect to both boundaries - i.e. the aforementioned confusion between B-PER and I-PER, as well as type-related errors, such as the mislabelling of B-ORG as B-LOC and even PER (confusion both boundary and type).

The second system exhibited an interesting pattern with regard to the categories. Firstly, the best precision scores occurred in the classes which included the B tag - i.e. named entities occurring at the beginning of a sequence. This is congruent with the fact that apart from O, the B-labelled classes are the majority classes in the distribution of the CONLL dataset. Furthermore, the combined feature system performed best in terms of precision for MISC, then PER, followed by ORG and LOC - this pattern occured also for the I-labelled classes. It would seem as though the dense features boosted the MISC class, which is relatively under-represented with respect to the other three categories PER, ORG and LOC. Some repeated errors included the boundary confusion of B-PER with I-PER (mentioned above), but also the reverse phenomenon - confusing I-PER with B-PER. This could be potentially fixed by focusing on features that are more specific to people, or even the incorporation of external gazetteers of person names, as was done for the CONLL 2003 task and many other NER tasks since. (Yadav and Bethard, 2018). Another type of persisting error was the confusion between B-ORG and B-LOC. In this case the system identified the boundary correctly, but mislabelled the type. By looking at some examples of mislabelled instances of B-ORG, this clearly happens because the first instance in an organization name is its geographic location or origin, i.e. 'China Steel' (a name of a company), combined with many additional instances of the same token tagged as B-LOC.

7 Discussion

When comparing the models to the baseline model, it seems as though the general performance in terms of both precision and recall is generally comparable for all models - all of which scored roughly around 70% for precision, and slightly less for recall (and subsequently, the f1-score which is determined by both the precision and the recall scores). The only exception was the SVM model trained with combined sparse and dense features, which performed significantly better than the rest - scoring 90% for precision and 89% for accuracy. This results are likely due to the combination of word embeddings, which provided valuable contextual information, alongside the solid baseline features which included the token, its part of speech, and information about word-case for the token and its preceding token. The features play a significant role in potentially improving precision and recall, and by performing feature ablation I was able to identify a pattern with regard to the different types of features combinations. Seemingly the absence of syntactic features for the SVM model augmented the number of false negatives quite significantly. At the same time, orthographic features relating both to word-case and grammatical properties of named entities had an effect on the precision scores, effectively reducing the amount of false positives. I find this relation particularly interesting, as it demonstrates that different linguistic aspects of words can have different effects on the learning process.

8 Conclusion

This paper featured an overview of a Named Entity Recognition task performed by experimenting with different models and features, including pretrained *word2vec* word embeddings. The models - Linear Regression, Naive Bayes, SVM and CRF were trained on the CONLL2003 datasets and evaluated by means of (macro-averages of) precision, recall and f1-scores. A detailed process of feature ablation, as well as hyperparameter tuning for the SVM-based system, is also brought in detail. Finally, an error analysis is conducted for two systems.

Acknowledgements

Thanks to Eva van der Heide for taking time to explain unclear concepts and shed light on the training process.

References

Hai Leong Chieu and Hwee Tou Ng. 2002. Named entity recognition: a maximum entropy approach using global information. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Asif Ekbal and Sivaji Bandyopadhyay. 2008. Development of bengali named entity tagged corpus and its use in ner systems. In *Proceedings of the 6th Workshop on Asian Language Resources*.

Ralph Grishman and Beth M Sundheim. 1996. Message understanding conference-6: A brief history. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Jing Jiang and ChengXiang Zhai. 2006. Exploiting domain structure for named entity recognition. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 74–81.

Jurgita Kapočiūtė-Dzikienė, Anders Nøklestad, Janne Bondi Johannessen, and Algis Krupavičius. 2013. Exploring features for named entity recognition in lithuanian text corpus. In *Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013)*, pages 73–88.

Uri Katz, Matan Vetzler, Amir Cohen, and Yoav Goldberg. 2023. NERetrieve: Dataset for next generation named entity recognition and retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3340–3354, Singapore. Association for Computational Linguistics.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

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.

Vikas Yadav and Steven Bethard. 2018. A survey on recent advances in named entity recognition from deep learning models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2145–2158, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Theoretical Component

What is Machine Learning?

Machine learning is concerned with finding patterns and structures in data in order to obtain information or insights, perform a certain task, or generate new data based on what is learned. The basic idea of machine learning is to develop systems that can learn from text, images or even sound, and generalize onto new, unseen input in a reproducible way. The goal is to accurately perform the task at hand without the need of a specific set of instructions. Machine learning can be in the form of a supervised process, whereby the learning is facilitated by preliminary queues (labelling) or feedback (reinforcement learning), or it can be an unsupervised process in which data is given without any additional information, and the system performs its own analysis to learn the structures within the data.

Sentiment Analysis

The 2013 SemEval Shared Task is concerned with sentiment analysis. The model used in the task is trained on a corpus of tweets and SMS messages. Subtask B in particular seeks to provide a sentiment label of positive, neutral or negative, given a single message. If the message contains more than one sentiment, then the aim is to identify the strongest one. Some useful features for this task include features pertaining to single words or tokens, such as positive or negative words sourced from external gazetteers or lexicons (i.e. great, good, terrible, nice), or n-grams or word clusters indicating sequences of positive or negative descriptions which can help in determining the sentiment when multiple instances of positive or negative words appear, or when negation is involved (i.e. terrible-movie, good-popcorn, not-great). Other features include syntactic roles, relations or dependencies (i.e. subject-object-predicate relations or pos-tagging), which can help in determining the subject of the message and focus the analysis there. Additional features may include case - for example, all-caps is used either for emphasis, or in acronyms conveying emotion such as LOL, LMAO. Punctuation such as exclamation mark, ellipsis, or full stop can help in conveying tone, or identifying emotion when used in emoticons. In order for the system to make use of these features, it is necessary to convert natural language to a structured, computable format. Vectors containing numerical coordinates allow for the representation of textual information as numerical information, which can then be used to compute probabilities or find patterns. To that end, the above features must be represented in a structured way. For example, a feature such as 'all caps' could be interpreted as a binary feature of 1 or 0, where the value is 1 if a word is in all-caps, and 0 when it is not. This information, which is represented to humans as word-shape, in the context of a particular language and its alphabet, could then be transformed into something computable, by representing world-dependent information as numbers. Such representations could be one-hot vectors with binary values, whereas more complex representations incorporating contextual information would be represented as more dense word embeddings.

Coreference Resolution

The task of coreference resolution aims to find all mentions that co-refer to the same entity in a given task. The idea is to create a link between a single entity, and the different manners in which it is referred to in the text. Such mentions can include anaphoras and cataphoras (words or pronouns referring to something that appeared earlier and later in a text, respectively), descriptions or job titles (e.g. the musician), or repeated references of the same word. Relevant features for this task could be syntactic categories such as part of speech - for example, nouns and pronouns are strong candidates for coreference. Syntactic relations such as subject-object-predicate are also useful for determining which pronouns refer to which nouns within a given sentence. Other features could be grammatical categories such as gender, number or person. The task of pronoun resolution, which shares similar objectives with the coreference resolution task, often looks at the grammatical gender of a pronoun in comparison to those of the names in a given sentence. Alongside the aforementioned syntactical and grammatical roles, other features utilized in the rule-based system of Lee et al (2013) were acronyms and demonyms which can help make connections between variations of names (e.g. United Nations and UN, or Israel and Israeli). The binary features such as part-of-speech or grammatical categories (i.e. gender, number and person) could be presented for the system as a feature vector or one-hot encoding with a binary value. In order to represent relations between words, we would have to combine features or take into consideration sentence structure or dependency parsing.

The representation of such relation would therefore be more complex than one-hot encoding. Once again, we convert linguistic and textual information into numeric representation to facilitate computations and transform human-readable information into machine-readable and computable values that will allow for generalization and identification of patterns.

K Nearest Neighbors

The K Nearest Neighbors algorithm (KNN) is a classification algorithm that is based on feature similarity. KNN classifies an instance in a database based on the majority class of its neighboring datapoints. In this method, a positive integer k is specified, and then the k nearest neighbors in the database are selected. Their proximity is determined by some distance metric, most often Euclidean distance. Then, based on the provided information in the database, these neighbors are classified, and then the most common classification among these is given to the instance. Choosing the correct k for the task is done by hyperparameter tuning. This is important, since if k is too small, then there is risk of overfitting, since only a small selection of instances from the database is observed, possibly overlooking wider trends. If k is too big, then underfitting may occur as the general majority class may be taken, neglecting feature similarity. Also, a large k can lead to higher complexity, resulting in high running time. Furthermore, the type of distance metric used may also affect the results.

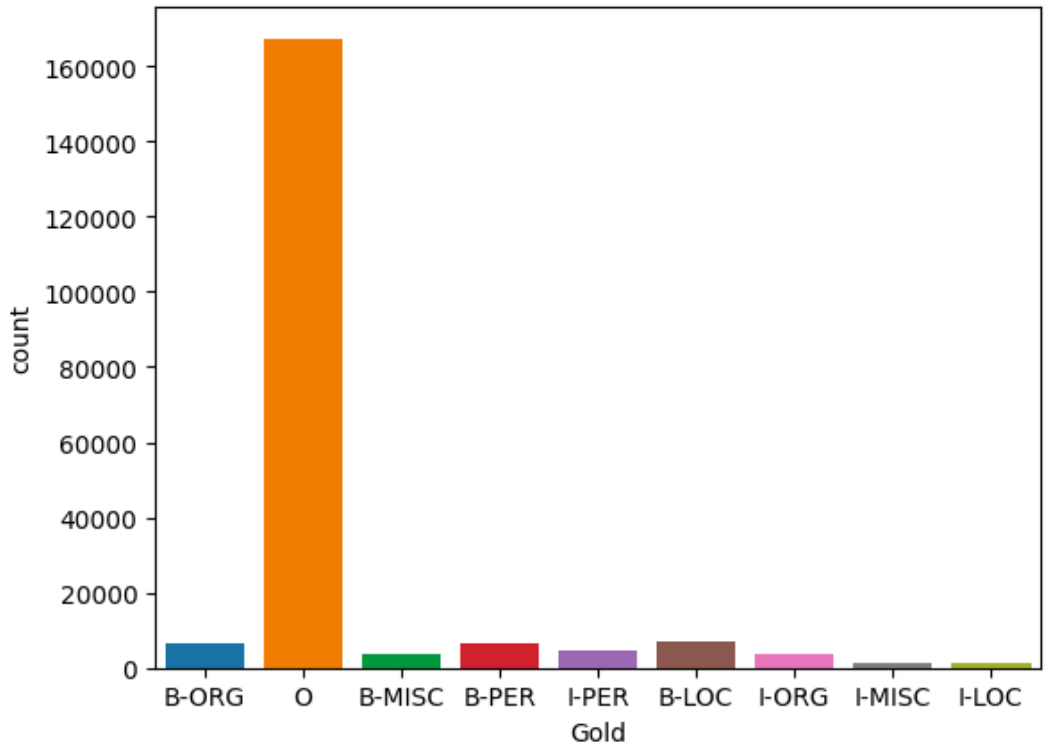
NERC-research preparation

The initial processing of the CoNLL2003 dataset consisted of displaying it as a dataframe using pandas, and then examining the features separately and combined. In order to assess the kinds of words belonging to each category, I created pairs of words and NER tags, and iterated through a list of pairs to count how many words appear per category. I created a dictionary in which the keys are the categories (i.e. B-PER, I-LOC etc.) and the values are lists of all instances. The number of instances was retrievable by getting the length of the value (list) of each key. I then created a dictionary of the most frequent words per category, setting a threshold of 20 as a frequency count, and retrieved the most frequent word, i.e. the word with the highest occurrence in the dataset. I printed the results in order to examine the most frequent words of each category manually, to gain insights and use these as a basis for feature design and extraction.

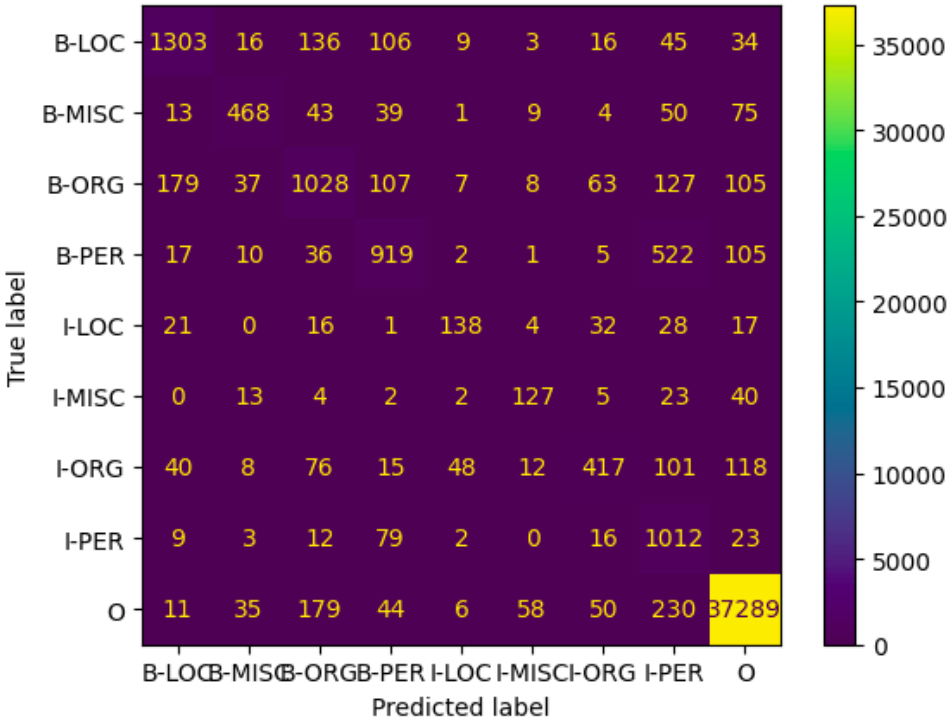
I evaluated my work in terms of precision, recall, and f-score, looking at their *macro*-average as a way to mitigate the significant skew towards the majority class O. When printing confusion matrices, I disregarded the class O so as to obtain more informative graphics with regard to the other classes. I mainly compared the precision and recall scores for each model. A high precision score is desirable for cases in which the risk of missing out is low, i.e. low-consequent tasks such as spam email detection, as we would be fine with a high number of false negatives - i.e. cases that were not returned by the system, even though they are relevant, if we have small number of false positives too. A system with high precision would be more hesitant to return a positive score but will be more effective, so it is prioritized when the price to pay when missing a relevant case is not too big. High recall is useful when there is a high risk of missing out - since we would like to minimize the amount of false negatives as much as possible. Examples for tasks for which high recall is important are flood or hurricane alerts, or cancer detection - cases in which the price to pay for not returning a relevant case and thus risk lives is too high. As for BIO tags, my evaluation looks for exact matches both in terms of type (LOC, ORG, etc.) and boundaries (B,I). I do not allow for partial matches, and I treat the span labels and classes as one combined metric. This was the metric used in the original CONLL 2003 shared task. As mentioned in §3.4, allowing for partial matches or disregarding the boundaries might be preferable in other tasks in which the goal is not to measure precision and recall scores of different systems and compare, but rather identify all the mentions of some named entity, allowing for flexibility in its detection process.

A Appendix

A.1 Distributions



A.2 SVM with baseline features: confusion matrix



A.3 SVM with combined sparse and dense features: confusion matrix