# Evaluation of Named Entity Recognition in Latin using an Unsupervised Model

**Gabriel Cristian Circiu**
IT University of Copenhagen
Copenhagen, Denmark
gaci@itu.dk

**Mykyta Taranov**
IT University of Copenhagen
Copenhagen, Denmark
myta@itu.dk

**Wenzel Keil**
IT University of Copenhagen
Copenhagen, Denmark
weke@itu.dk

## Abstract

Tackling the challenges of Natural Language Processing (NLP) in Latin using an unsupervied model has lead to the observation that certain tokenizers such as BERT are better suited for such tasks and that the choice of the model itself plays a significant role in the final results. Clustering has proved to be the most straightforward method to group entities together but a more intricate pipeline lead to a significantly better result, achieving an F1 score of !!!INSERT F1!!!.

## 1 Introduction

As international students, the probability of sharing a common language outside of English is not high, however, during our studies we came to the realization that some of usshare knowledge in one such language, Latin. Albeit we are nowhere near fluent, it has sparked curiosity to work with it. We chose to head in the direction of our shared curiosity of Latin, and to poke at it within the Named Entity Recognition (NER) sphere.

As such we chose to build upon a past research, titled *Challenges and Solutions for Latin Named Entity Recognition*[1] (Erdmann et al., 2016), which tackles the problem of NER in Latin using a supervised and semi-supervised model. Our dataset is based on the one used in the original paper, following the same structure, with some additional specifications for clarity.

## 2 Dataset

The dataset from the original paper named 3 works of literature, Caesar's *De Bello Gallico (BG)*, Pliny the Younger's *Epistulae (EP)*, and Ovid's *Ars Amatoria (AA)*. We have had the great fortune of receiving the full dataset in its entirety, as large text files, in IOB format. While all the literature was scrambled in each file, having it in the right format has made it all the more easier to process, and do some Exploratory Data Analysis (EDA). After thorough investigation, we have found that there is a slight inconsistency in the original paper regarding the dataset, and that of which we have received. As mentioned, the entirety of *BG*, and parts of *EP* and *AA* were annotated, however, parts of Caesar's *De Bello Civili (BC)* were also annotated within the dataset, but left unmentioned in the original paper.

To have an initial understanding of the expectations of our models, we must understand the distribution of the dataset, namely the amount of *Named entities (NE)* and *Non-Named entities (Non NE)* in the dataset. As shown in Figure 1 and Figure 2, we can see that there is a significant imbalance in the dataset, with only about 5.5% and 3.3% of the tokens being NE. The distribution of the NE's are fairly balanced beween the different types of entities.

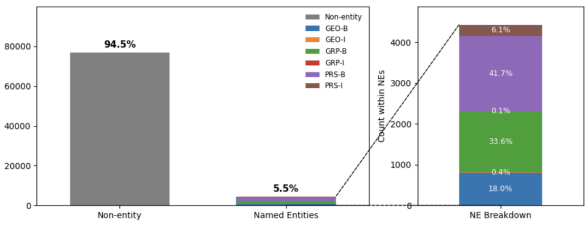


Figure 1: Distribution of data for *BG*, *BC*, and *AA* combined, used for one of the training folds.

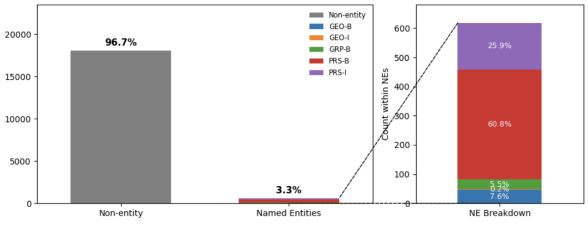


Figure 2: Distribution of data for *EP*, used for one of the testing folds.

[1]Original paper: https://aclanthology.org/W16-4012/

## 3 Methodology

Once we had the dataset split into training and testing files, we started with tokenizing the words, and finding a solutions on how to handle the words, embeddings, and labels, without having supervision over the original labels that we were given. Afterwards we have decided on multiple approaches in training a model, namely Clustering, and a Neural Network.

### 3.1 Embeddings

Our initial approach was to vectorize tokens using a basic `Word2Vec` method from the `Gensim` library. However, this produced unacceptably low results with clustering — the model wasn't able to separate entities from non-entities in a meaningful way. So we moved on to a more powerful method: using contextual word embeddings from a `pretrained Bidirectional encoder representations from transformers (BERT)` model, specifically `bert-base-multilingual-cased`. This model was trained on a large corpus of multilingual text, including Latin, and is capable of generating high-quality embeddings for words in context.

We used the `Hugging Face Transformers` library to load the model and generate embeddings for our tokens. We extacted the embeddings from the last hidden layer of the model. Since `BERT` works on subword level, we combined the embeddings of the subwords that make up a token by averaging subwords embeddings. This approach has been shown to work well in practice and allows us to obtain a single embedding for each token. We discusse the options of using subwords embeddings for clustering and switching to the word level later on, but this led to unnecessary complications of the pipeline, for example, by introducing the necessaty of handling scenarios when embeddings of one word end up in different clusters.

### 3.2 Clustering

Once we had the word level embeddings, we applied `KMeans Clustering` to automatically group the tokens into clusters. `Kmeans` was the most straightforward choice for the task. One of the challenges we faced was the choice of `K` - number of clusters. We tried several methods to determine the optimal number of clusters, including the *Elbow Method* and *Silhouette Score*. The elbow method suggested that big values of `K` are better, up to the point that number of clusters was close to the number of tokens. Silhouette score also was not very helpful, since bigger score didn't correspond to better evaluation scores afterwards. So we boiled down the problem to the simplest approach: trial and error. We ended up with !!!INSERT K!!! clusters.

We used the KMeans implementation from the `scikit-learn` library, which provides a simple and efficient way to perform KMeans clustering. After clustering we named each cluster by taking the most frequent token in the cluster. We had a short discussion on whether this makes the model semi-supervised, but concluded that it doesn't: the label data is only used after clustering, and doesn't influence how the clusters are formed. Then we evaluated the clusters as a baseline for our model. We used the F1 score to measure the perfomance, and got a score of !!!INSERT F1!!!.

### 3.3 Neural Network

Next we reconstructed the model using a `Neural Network` (NN). We used a lightweight NN consisting of two linear layers with `GELU Activation` and `LayerNorm`, mapping contextual BERT embeddings to entity class logits. The network is trained using pseudo-labels derived from the initial clustering, which makes the inherently supervised neural network **unsupervised**.

We used `Pytorch` to implement and train the model. The choice of the number of epochs was based on the rate of change of the F1 score, which we observed in most cases to not significantly increase after 10 epochs, and plateau after 30 epochs. As shown in Figure 3,

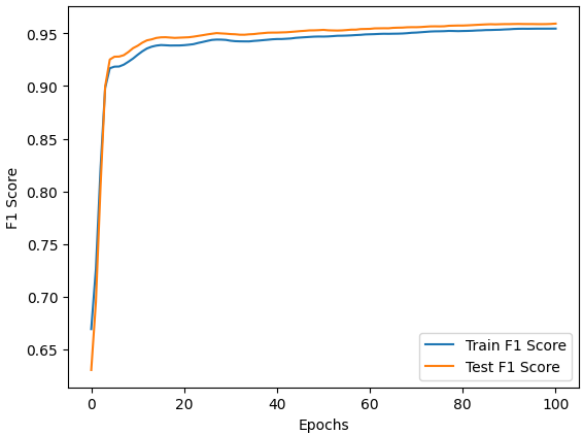


Figure 3: Rate of change of F1 score with increasing number of epochs.

As such, we stopped at 30 epochs.

## 4 Results

Notes: Binary Clustering, Multi-Class Clustering, and Neural Network.

Our results:

| Command | Output | Command | Output |
|---|---|---|---|
| {\"a} | ä | {\c c} | ç |
| {\^e} | ê | {\u g} | ğ |
| {\`i} | ì | {\l} | ł |
| {\.I} | İ | {\~n} | ñ |
| {\o} | ø | {\H o} | ő |
| {\'u} | ú | {\v r} | ř |
| {\aa} | å | {\ss} | ß |

Table 1: Example commands for accented characters, to be used in, *e.g.*, BibTEX entries.

## 5 Future Work

As our results are not very promising, it gives us a glimpse into the mysteries of the Latin language, and the challenges of NER within it. We plan to explore some aspects of this field as there is a lot of room for improvement, and a lot more to discover.

To start, we would like to explore having a much larger dataset to work with, and observe how results improve with the proportion of data. Next, we would like to explore different capitalizations of the language, and observe how results change, as we suspect that the capitalization played a significant role in the performance of the model. Finally, considering an ensemble method of different models may prove to be the most effective approach, as it has been observed so far, and we would like to explore the limits of this approach and benefits of it.

## Limitations

Due to the size of the dataset and the complexity of the task, we were not able to train a more capable model. A similarly comparable evaluation of NER using an unsupervised model within the English language has been shown to achieve much better results, indicating that the size and distribution of the dataset is a significant factor in the performance of the model, even more so than the choice of model.

Having a discrepancy between the original dataset wordcount and what we have received, based ont the original splits, having to manually split the dataset has made our work substantially longer than expected, as we had to manually separate the dataset up into new files. Using the help of the original source of the dataset, we managed to find and split it appropriately, at a significant cost of time.

Due to the limitations of time, we were not able to explore more methods of approach and tinker extensively with the hyperparameters of the model.

## References

Alexander Erdmann, Christopher Brown, Brian Joseph, Mark Janse, Petra Ajaka, Micha Elsner, and Marie-Catherine de Marneffe. 2016. Challenges and solutions for Latin named entity recognition. In *Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH)*, pages 85–93, Osaka, Japan. The COLING 2016 Organizing Committee.