Lab deliverable 2 : NERC task

Mining Unstructured Data, Master in Data Science, Barcelona School of Informatics

Javier Herrer Torres (javier.herrer.torres@estudiantat.upc.edu) Ignacio Lloret Lorente (ignacio.lloret@estudiantat.upc.edu)

GitHub repository

Introduction

This report presents an exploration into Named Entity Recognition and Classification (NERC) tasks using machine learning techniques. The primary goal of this work is to develop a system capable of identifying and categorizing drug names within unstructured text data.

The project focuses on implementing a Sequence Tagging approach using the B-I-O schema.

To accomplish the NERC task, the project adopts a structured approach consisting of feature extraction, model training, and classification. Feature extraction involves capturing relevant information about tokens and their contexts to inform the learning process. Various machine learning algorithms are explored for model training, including Conditional Random Fields (CRF) and Naive Bayes, with an emphasis on experimenting with different parameterizations and feature combinations to optimize performance.

For this we will play with different features extracted from the tokens in order to accomplish this task. We will also try different algorithms to try to improve the CRF model. We will mesure the improvements made by the feature extractor by predicting a test dataset containing entities that the model must predict its category. As the main metric we will use is the recall and F1 in order to measure how the model performs in an unbalanced target challenge.

The deliverables of this project include a comprehensive report detailing the methodologies employed and insights gained from the experimentation process. Additionally, the report includes the extract features function, which forms a critical component of the feature extraction process, along with any relevant code snippets.

Overall, the objective of this project is to develop a proficient NERC system for drug name identification and classification, leveraging machine learning techniques to achieve accurate and efficient entity recognition within unstructured text data and test the importance of feature extraction in the context of NERC.

Machine learning NERC

After experimenting with multiple algorithms including Gradient Boosted Trees, Multinomial Naive Bayes, and Random Forest, we found that none of them could match or surpass the performance achieved by the Conditional Random Fields (CRF) for this particular task. While these alternative models achieved F1 scores of 25, 49, and 18 respectively, with Naive Bayes being the most successful among them, they still fell short compared to the CRF.

Given the superior performance of the CRF, we decided to focus our efforts on optimizing its hyperparameters rather than switching to a different algorithm. We explored various hyperparameters, including different

algorithms such as L-BFGS with L1/L2 regularization, SGD with L2-regularization, Averaged Perceptron, Passive Aggressive, and Adaptive Regularization of Weights (AROW), as documented in the Documentation CRFSuite. However, none of these alternatives provided an improvement over the initial SGD with L2-regularization.

Additionally, we fine-tuned the regularization parameter in an attempt to enhance the accuracy of our CRF model. Although we made a slight adjustment by reducing the regularization parameter from 0.1 to 0.08, we did not observe a significant difference in performance.

Changing the mininum frequecy of a feature did not improve the performance of the model.

```
# Use L2-regularized SGD and 1st-order dyad features.
trainer.select('l2sgd', 'crf1d')

# This demonstrates how to list parameters and obtain their values.
trainer.set('feature.minfreq', 1) # mininum frequecy of a feature to consider it
trainer.set('c2', 0.08)           # coefficient for L2 regularization previously
to .1

print("Training with following parameters: ")
for name in trainer.params():
    print (name, trainer.get(name), trainer.help(name), file=sys.stderr)

# Start training and dump model to modelfile
trainer.train(modelfile, -1)
```