澳門大學
UNIVERSIDADE DE MACAU
UNIVERSITY OF MACAU

Faculty of Science and Technology

CISC3025 – Natural Language Processing

Project 3: Implementation of a Maximum Entropy Model for Named Entity Recognition

Group Members:

Huang Yanzhen DC126732
Chen Zirui DC127901

Table of Contents

1 Introduction

Being a part of the syntax level processing techniques in NLP, Named-Entity Recognition is a mechanism for a subtask of information retrieval that's been applied in various fields, such as search engines.

To be more specific, Named Entity Recognition involves assigning a class label for entities within a piece of plain text. Those classes could include person names, firm or organization names, locations, dates, and more customed tags as you want. This project made use of the Maximum Entropy Classifier from *scikit-learn* to implement a simple NER task, which is to recognize person names.

The initial task for building a MaxEnt model is to select proper features. Once the features are selected, the training set, which are the word tokens pre-assigned a label, are passed along with the features to train the model. The model will assign an importance weight to each feature for each class. Lastly, to test the model, for each entity, the model calculates the vote of each class using the features and the weights, finally deciding the class label.

In this project, the training part is done by the encapsulated model in *scikit-learn.* The main workload of this process is to process training data to select features, and to build a website that demonstrates the performance of the model, which runs at the backend in the local server.

2 Methods and Implementation

2.1 Features Selection

Feature selection aims to find representative features of a word that could produce the maximum entropy of the classification, that is, to best distinguish the classes. There are three types of features, which are Internal Pattern Features, Library Features, and Contextual Features. For the binary case, there are positive and negative features. Positive features indicate that a word is more likely to be a human name. Negative features, on the contrary, unrecommends to assign the label of human name to the word.

2.1.1 Internal Pattern Features

Internal Pattern Features put their scopes in the word itself. It analyses the patterns of a word and assumes that some specific pattern can distinguish a person's name from others.

Table 1. Internal Pattern Features

	Feature Name	Feature Description	Reason for selection	Match Examples
+	p_cap_low	Start with Capital letter, and the rest letters are lowercase. There may be a prime after the first capital letter. There may be a second capital letter in the third letter's space.	In English, names always start with a capital letter. There are also some special name styles, like MacArthur, and O'Brien.	Jenny, MacArthur, O'Brien
+	p_cap_period	A single capital letter followed by a period.	In English, human name initials	D., J., M., …
-	p_noun_like	Have some noun-like ending patterns, like *-tion, -sion, -ance,* etc.	Human names doesn't always ends with the same pattern as some special nouns that are derived from verbs or adjectives.	mention, motion, addiction, expansion, allowance, …

The corresponding regular expressions of these features are as follows.

	Feature Name	Regular Expression
+	p_cap_low	^[A-Z](\'[A-Z]?|[a-z][A-Z])?[a-z]+
+	p_cap_period	^[A-Z]\.$
-	p_noun_like	(([aio]?tion|ment|ness|ship|hood|\w+age|[ae]nce)[sd]?$)/i

Table 2. Regular expressions for internal patterns

2.1.2 Library Features

Library Features cares about whether a word belongs to a corpus whose members are likely or not to be classified as names. For example, the word "Tuesday" belongs to the corpus of "Dates" and the word in corpus of "Dates" is not likely to be tagged as a name.

There are a pack of words that is likely not to be classified into human names. For instance, week names like *Monday* and *Tuesday* are not likely to be human names. Country names, like *Korea, China, Japan* or *Britain* are also not likely to be classified to human names. Therefore, we added some libraries that contains these kinds of words and combined them into one evaluation criteria. That is to say, as long as a word belongs to one of the libraries in this set, it will be assigned a negative feature. The negative library set includes the following.

Table 3. Negative feature libraries

Libraries	Python Packages
Week Names	*self-defined*
Month Names	*self-defined*
Country Names	*geonamescache*
City Names	*geonamescache*
Stop Words	*nltk.stopwords*

On the opposite side, there are pack of words that are likely be classified into names. One of the most important packs are, exactly, names. Therefore, an *nltk* corpora is used to identify whether a token is always presented as names. Likewise, if a token belongs to this library, it will be assigned a positive feature.

2.1.3 Contextual Features

Contextual Features looks beyond the word itself, and seeks evidence in the contextual environment where the word is in. For instance, a name tends to be in the first and second place of a sentence. A name also tends to be in the main reference of an attributive clause. Unlike Library Features, Contextual Features are discrete, that is each pattern matches only one feature, unlike in Library Features, one feature is matched to numerous patterns.

Table 4. Contextual Features

	Feature Name	Regular Expression
+	is_start_of_sentence	$i = 0 \wedge w_{i-1} = "."$
+	is_target_of_clause	$w_{i+1} = "." \wedge w_{i+2} = "who" \vee "whose"$
-	is_after_status	$w_{i-1} = "Mr." \vee "Ms." \vee "Mrs." \vee "Dr." \vee "Prof."$

There are some trivial contextual features recorded in the model training function demonstrated in the form above. For instance, a name is highly likely to occur at the start of a sentence. A name is also highly likely to occur after social statues, like “Mr.” and “Ms.”.

However, compared to the two mentioned above, the most useful contextual feature is that, the target of a restricted attribute clause is always a human name. For instance, “Joe, who was the president…” is an attributive clause and Joe is a human name. The attributive clause puts a high probability on its target to be recognized as a human name.

2.2 Server Implementation

We selected *Django* as our backend framework, since it is written in python and thus have better compatibility to our model. The front-end is written in plain HTML and styled using plain CSS, for simplicity. Additional JavaScript is written to instantiate the XML HTTP requests, aiming to submit the input query and handle the JSON response from the backend and display the results in the browser without leaving the current page.

Django is a web app framework using Model-Template-View software design pattern. Our NER model, in this case, makes use of the *View* component. Given a URL, it accesses the backend resources in the corresponding view.

Below shows the abstract directory hierarchy of this project for better reference.

Code 1. Directory hierarchy of this project

```
CISC3025-Name-Entity/
|- name_entity_server/          --> Server directory
   |- name_entity_server/       --> Default app of this server
   |- NER_app/                  --> Custom app of server, in this case, the NER app
      |- migrations/
      |- NER/                   --> Project Main Directory, i.e. from Moodle
         |- data
         |- __init__.py         --> For pre-downloading nltk corpora
         |- MEM.py              --> MaxEnt Model
         |- playground.py       --> Prediction function, connect model with backend.
         |- run.py
      |- templates/
         |- nerUI.html          --> Webpage UI
      |- __init__.py
      |- views.py               --> Defines the JSON response of backend.
   |- __init__.py
   |- manage.py                 --> To start server
```

To run the server, it is required to change directory into the first `name_entity_server` directory. After that, run command `python -m manage runserver` to invoke the manage.py file, running the server. More instructions and exception handling is described in the `Readme.md` document in the repository.

2.3 Frontend-Server Connection

2.3.1 Prediction Function

Prediction is performed using the dumped model. A prediction function is required to perform the actual Named Entity Recognition task. This function takes an input of a sentence written in natural language and adds label to name entities that it recognizes as a name. Below shows the algorithm of this function.

Algorithm 1. Predict entities of person names in a sentence

```
Procedure PredictEntities(sentence, classifier, features);
Let tokens ← SplitWordsWithPunctuation(sentence);
Let prevLabel ← "O";
Let predictedLabels ← ⦰;
For position in 1, … tokens.length do
    curLabel ← classifier.classify(features);
    prevLabel ← curLabel;
Output predictedLabels, tokens.
```

2.3.2 Overall Structure of Project

The frontend JavaScript and the backend Django server works together to perform the task. Their work could be divided into three steps.

First, the browser passes an XMLHTTP request to the server, carrying the input sentence.

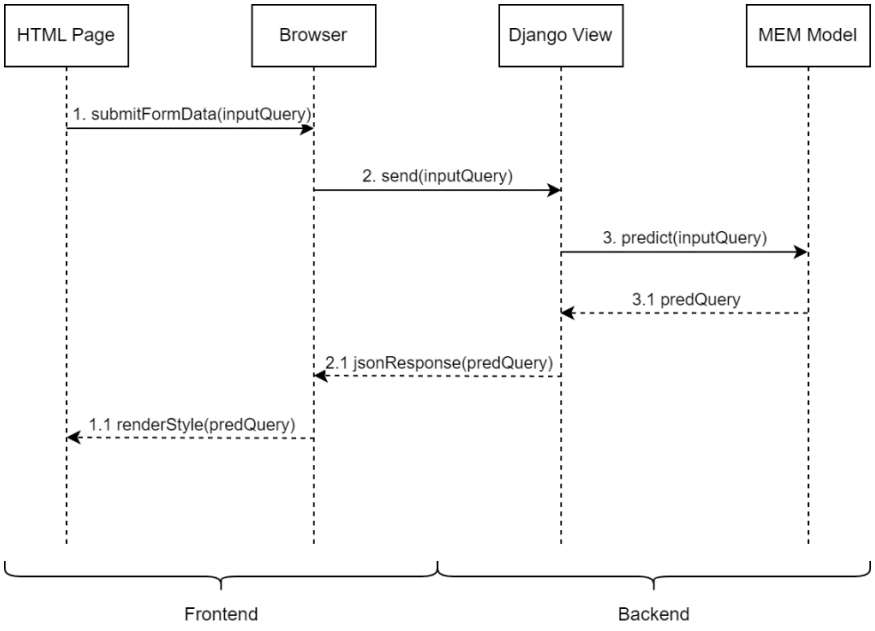
Second, the input sentence is passed to the corresponding view defined in Django specifically for the task of named entity recognition. In the view, the MaxEnt model is invoked and the *PredictEntities* function is used to parse the sentence and label the entities. The view would also add a span tag to indicate the browser to highlight the entities that's been classified as a person's name. The span tag is demonstrated below.

```
<span style="background-color: #bce7ac">
```

Lastly, the view would pass the parsed sentence back to the browser using JSON response. JSON response is better for native web designs since it allows us to only change the inner HTML content of the pre-set paragraph space instead of switching multiple webpages.

Below shows the UML Sequence Diagram that denotes the overall working principles of this frontend-server structure.

Figure 1. UML Sequence Diagram of the communication principles of the frontend-server structure



3 Evaluations and Discoveries

3.1 Issues Discovered and Fixed

3.1.1 Over-Fitting Problem

Initially, way more negative features than the present ones were applied, including the internal pattern features of All capital letters, Numeric expressions, Location name abbreviation, etc. Also, the library features and are very dispersed. Too many features caused an over-fitting problem, as we observed that our first version of the model is only able to recognize the name of "Johnson". Below shows some discarded features.

Code 2. Initial Version of our code

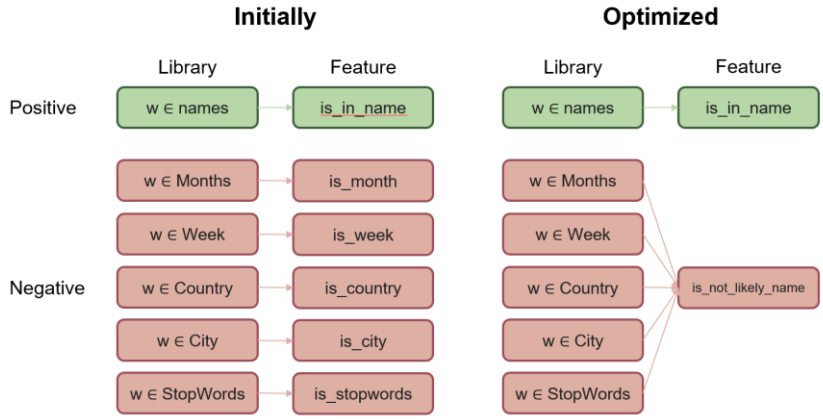
```
- Person status prefix.      e.g. Mr., Ms., Mrs., ...
'p_name_prefix': re.compile(r'M[a-z]{1,3}\.'),

- All capital letters.
'p_all_cap': re.compile(r'^[A-Z]+$'),
# - Possessive case. e.g. 's, ....
'p_possessive_like': re.compile(r'\'s$'),

# - Location name abbreviation. U.S., U.K., D.C.,  ...
'p_country_abbrev_like': re.compile(r'([A-Z]\.){2,3}'),

# - Numeric expressions.
'p_num_slash': re.compile(r'(\d+-)+\d+|\+\d+|\d+|\d+\.\d+')
"""
if previous_label == 'PERSON':
    features['is_previous_person'] = 1

if previous_label == 'O':
    features['is_previous_other'] = 1

# ------------- Position Related -------------
Is around the first place in a sentence.
if (position > 0 and words[position-1] == ".") or (position > 1 and words[posit
    features['is_around_first'] = 1

Is the last word
if position == len(words) - 1:
    features['is_last_word'] = 1

# Is after name prefix
if position < len(words) - 1 and re.match(r'M[a-z]{1,3}\.', words[position-1]):
    features['is_after_name_prefix'] = 1

# + Is in possessive case
if position < len(words) -1 and words[position+1] == "'s":
    features['is_possessive'] = 1

if words[position+1] == "verb":
    features['is_after_verb'] = 1

if words[position - 1] in stop_words:
    features['is_after_stop_word'] = 1

if not position >= len(words) - 1 and words[position + 1] in stop_words:
    features['is_before_stop_word'] = 1
"""
```

To cope with this problem, we brought up some improvements about this model.

First, we realized that there were too much negative tags in the Internal Pattern Features set and Contextual Features set. We thus cut most of them since we think is not quite useful in distinguishing entity classes.

Second, we merged multiple library features. At first, one library is matched with one feature, i.e., there is a one-one relationship between the set of libraries and features. However, we realized that negative library features work bad individually in the process of distinguishing, despite it produces a high F-Score. That is to say, instead of working in a dispersed manner, working together to filter an entity step-by-step to finally to conclude only once whether one word is or is not likely to be a name is robust enough and need no further breaking. The below figure demonstrates the merging process.

Figure 2. Merge library features



By performing these optimizations, the performance of our model met a breakthrough. Although the recall value was slightly lower, the model recognizes more names than before.

3.2 Advantages and Drawbacks

3.2.1 Advantage: Good Performance on Attributive Clauses.

Our model works very well in recognizing the feature of restricted attributive clause targets. Figure 1 demonstrates a great recall performance that most names are recognized. Figure 2 shows a comparison that, applying additional attributive clauses produces excessive performance to the model.

Figure 3. Works well in an attributive clause

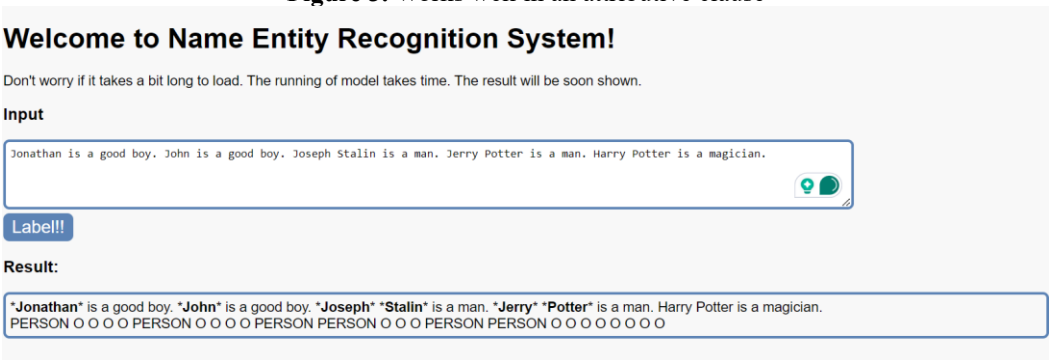
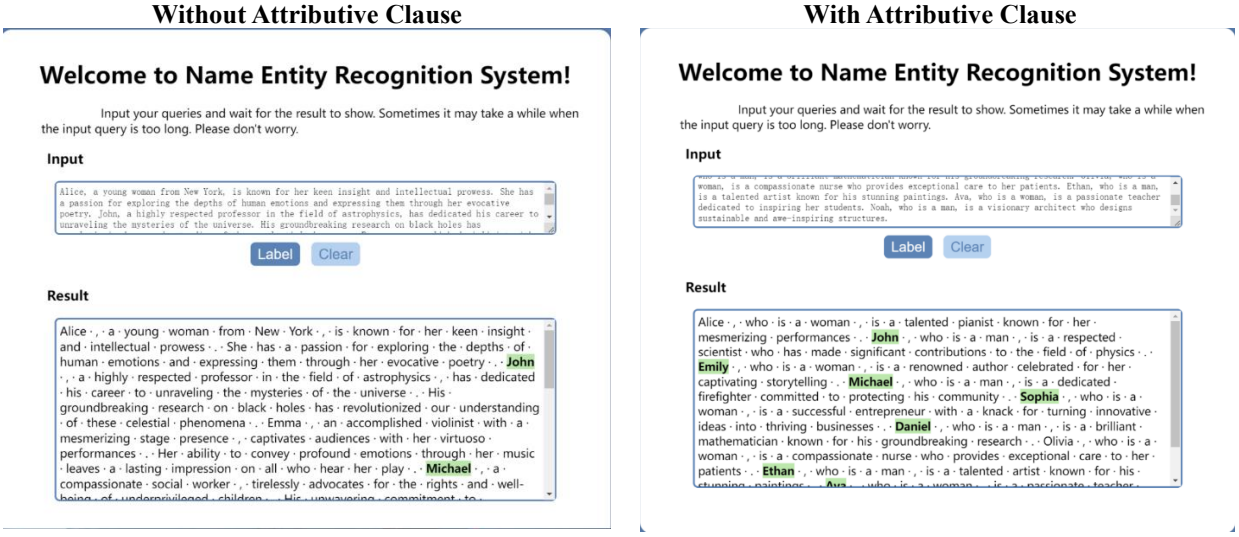


Figure 4. Good impact of adding attributive clauses to sentence

Without Attributive Clause **With Attributive Clause**

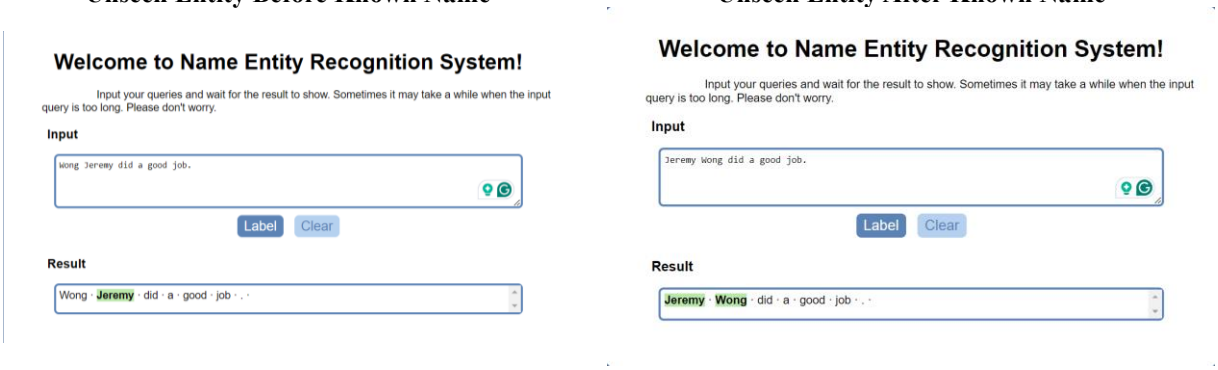


3.2.2 Drawback: Over-Reliance on Known Data

The model has a reliance on the name corpus whose extent is higher than our expectation. Below shows a concrete example of a phenomenon that, how the position of an unexpected entity results in the accuracy of the prediction. It is obvious that unseen names can only be recognized if they are after the names that's seen in the corpus.

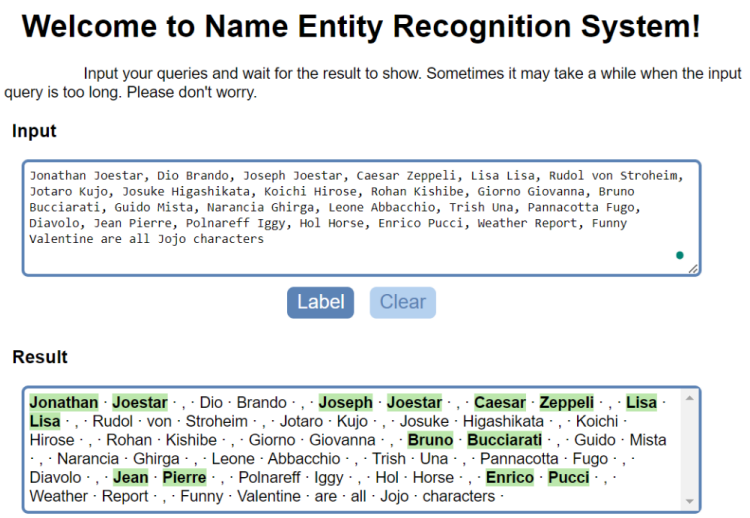
Figure 5. Comparison of the position of unseen text

Unseen Entity Before Known Name **Unseen Entity After Known Name**



Another example of the effect of this over-reliance is that the model is not quite friendly when it is faced with non-English names. We performed a stress test on the corpus of JoJo movie character names and found that the recognition rate of Japanese and Italian names is much lower than the English ones.

Figure 6. Model Performance on JoJo movie character names



However, by removing the name corpus from the trained model, we faced a trade-off that the accuracy drops from around 9.8 to 9.7, and the performance of the model had a serious discount. In this case, only contextual features can be recognized, and the model displayed another over-reliance on the training dataset. In other words, it tends to recognize the existing names in the data set and tends not to recognize unseen entities. Therefore, we finally decided to keep the original model.

4 Conclusion

In this project, we performed three major tasks on the Maximum Entropy modelling for Named Entity Recognition tasks, which are feature selection, frontend-server construction, and model testing and evaluation.

First, we selected multiple features that we think is robust to distinguish between two classes “O” and “PERSON”. The features are divided into three categories, which in order are Internal Pattern Features, Library Features, and Contextual Features. Internal Pattern Features focuses on the pattern or shape of a word entity. Library Features concentrates on whether a word entity belongs to a corpus that is likely or not to be classified as a name. Contextual Features focuses on the external environment in the word name. One great contextual feature is to observe words that’s a target of a restricted attributive clause.

Second, after we trained the model using the selected features, we wrote a prediction function that takes an input of a sentence and outputs the corresponding human names. We wrote a webpage that’s connected to a Django backend for better user experience.

Lastly, we used the user interface to test the model. We conducted multiple tests including stress testing and multi-source testing. We also observed the extrinsic evaluation data of this model, including F-Score, Precision, and Recall every time we retrained the model. We observed that high F-Score doesn’t mean better performance and over-fitting issue indicates that it is better to select features in a more careful manner.

5 References

Borthwick, A. E. (1999). *A maximum entropy approach to named entity recognition* (Order No. 9945252). Available from ProQuest Dissertations & Theses Global. (304524774). https://libezproxy.um.edu.mo/login?url=https://www.proquest.com/dissertations-theses/maximum-entropy-approach-named-entity-recognition/docview/304524774/se-2