Software architecture specification for Named Entity Recognition Software in Web News Articles

(SSL Project 20 April 2016)

Team : Dan Bizdadea, Florentina Bratiloveanu, Alexandru Catalin Ciobanu, Mihaela Sorostinean

1 Introduction

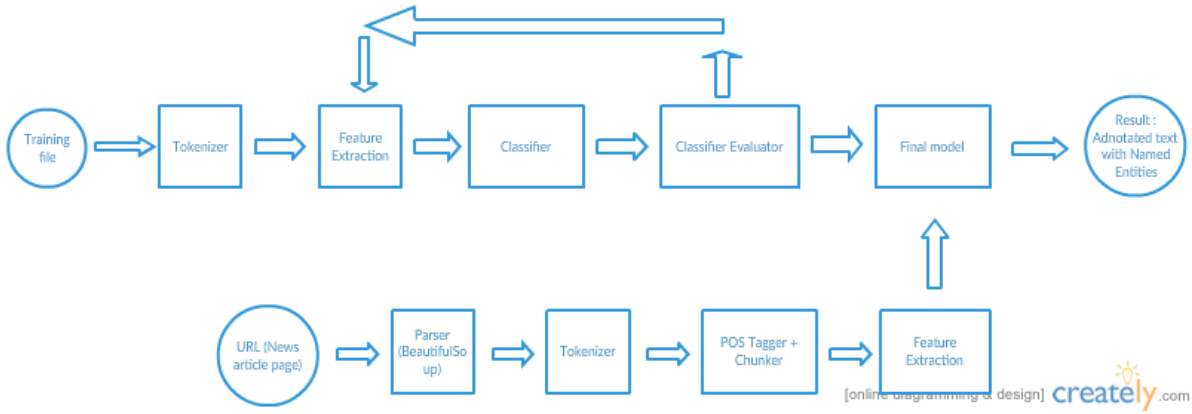
The Named Entity Recognition Tool will be developed in Python using scientific libraries like NumPy, scikit-learn and Pandas and natural language processing tooling (NLTK).
The scope of the project is to be able to parse news articles from the Web and return text annotated with named entities for People, Locations and Organizations.
The project is an intermediate step for a larger project that will attempt to find relations between different entities appearing in a text using simple queries like SQL.

2 Challenges

The most complex aspect of the project is successfully recognizing named entities. The difficulty of the task comes from the ambiguity of the English language where words like Baker can have different meanings like person name, occupation, organization (Baker Co) or location (Baker Street)
One way to diminuate this problem is to use chunking which segments and labels multi-token sequences.

3 Overall Architecture

The basic software architecture of the software is described in the diagram below:



Here is a small description of the components and workflow :

· **Training file** : This file contains the training data from conll2002 for English text. It is in the IOB format expanded with NE tag. An extract is presented below :

```
The DT I-NP O
European NNP I-NP I-ORG
Commission NNP I-NP I-ORG
said VBD I-VP O
```

· **Tokenizer :** This component will split the text into lists of sentences. Each sentence will contain a list of tokens. Each token is a list of 4 elements (word, POS Tag, Chunk Tag, NE Tag)

· **Feature Extraction :** This component will use the information from the list of sentences in order to create as many features as possible that can be used later for classification. Example of features : pos tags for words before, pos tags for words after, has initial capital letter, contains spaces, etc. This component will return 2 lists: one for the training set and one for the test set

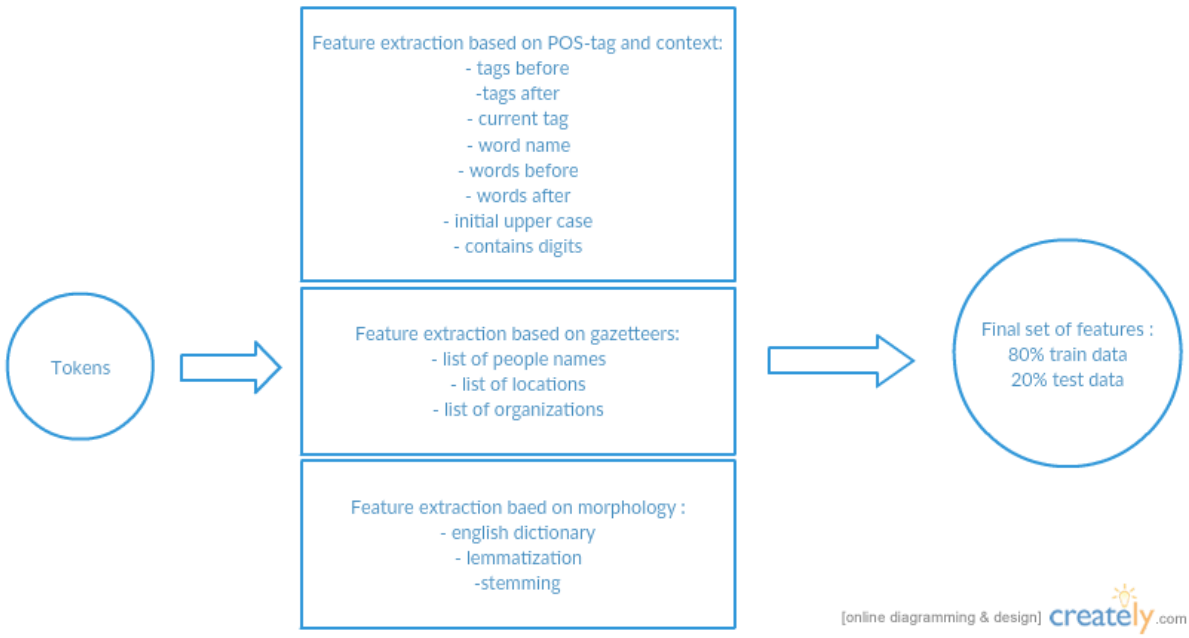
· **Classifier :** This component will feed the features to different classifiers. In order to do that, for each classifier there will be a feature selection and normalization step in order to improve the accuracy of the respective classifier.

· **Classifier Evaluator :** This component will evaluate the results of the classifiers against the test set. It will store the best solution that will represent the final classifier to be used on real data.

· **Final model :** This will be the final classifier selected for the job (It can actually be more classifiers combined) It will be used for real world data.
· **URL (News article page) :** This will be the source for the real world data. We'll decide on a particular news site : the guardian, cnn, nbc, etc
· **Parser (BeautifulSoup) :** this will download the html for the article page and will remove the HTML from text. It will return a raw text to the next component
· **Tokenizer:** This will split the raw text into sentences and words and send the list to the next component
· **POS Tagger + Chunker** : This will use the Python NLTK tools to perform pos-tagging and chunking on the text. It will send a text in the IOB format to the next component
· **Feature Extraction** : This is the same component used for training, this time without the 4th column (NE Tag). It will return the same list of features as before but without the target class associated and will send it to the final classifier
· **Result** : In the end, the final classifier will use the features obtained from real data article and return the same text annotated with named entities. This text will be displayed to the user.

3.1 Feature extraction

The feature extraction component will be the most important in the classification process, because it will need to gather all the necessary information from text. This is why we are going to cover this component in greater detail here.
As for the overall architecture, we'll start with a diagram of the component and we'll explain each sub-component :



The Feature Extraction component will receive the tokens as a list of sentences from the training data and will return 2 lists : one for the training set covering minimum 80% of the data and one for the test set covering maximum 20% of the train data.

There are 3 sources that we are going to use in order to build our features :

1. Feature extraction based on POS-tag and context:

Here we are going to make use of the current tagging of the words in order to extract patterns from syntactic rules. In order to do this, we'll create features based on :

- Current word
- Previous words
- Next words
- Current POS Tag
- Next POS Tags
- Previews POS Tags
- Previews NE Tag
- Next NE Tag
- Word suffixes of length 1, 2, …, N
- Word prefixes of length 1, 2,…, N
- Trigger words (nouns and verbs)
- Capitalisation of current word
- Information about having digits or other special characters inside
- etc.

Note : Considering the large number of words in the training data, if we are going to use binary features, for the prev and next words features we are going to include thousands of new binary features. Considering the large number of potential features, we included the feature selection component that will be described in the next section in order to diminuate the list to the most relevant one.

2. Feature extraction based on gazetteers :

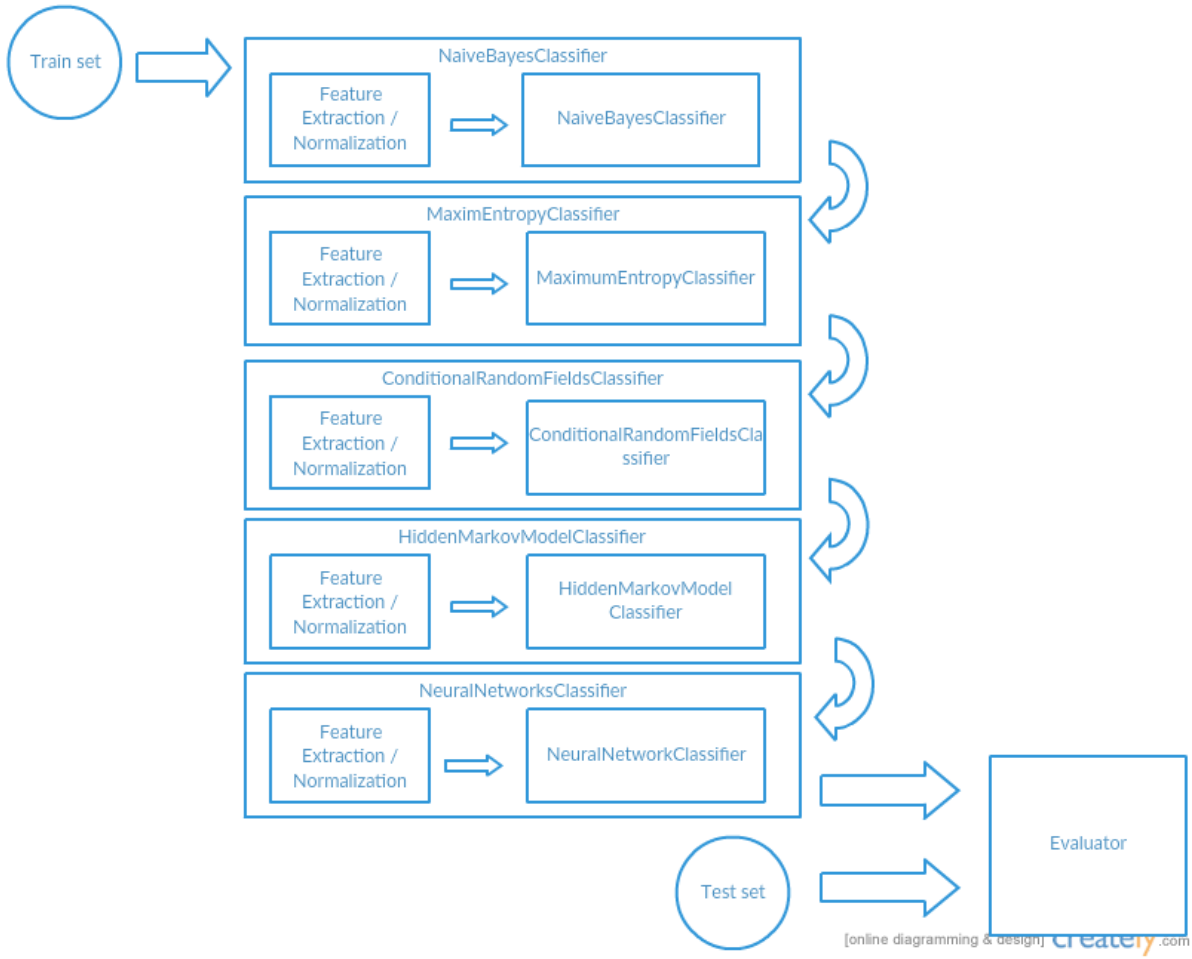
Here we are going to use as many resources as possible related to names of geographical locations, names of people, names of organizations. For each of them we'll create a binay feature that will state if the current word is part of any of the gazetteers described.

3. Feature extraction based on morphology :

Here we are going to use a little more information about the current word. We'll use WordNet lemmatization to find out if the word is part of the English dictionary. This can be useful considering that most named entities do not appear in the dictionary of a language. Stemming can also be useful here. Normally named entities do not have multiple forms (ex: plurals), so if there are dictionary words that reduce the the current word stem we can consider this as a feature meaning that it's probable that the word is not a named entity.

3.2 Classifier and evaluator

The next most important component of the system is the classifier, or more exactly the evaluation of classifier and selection of the best classifier/classifiers that can be used in the end.
The idea for this component is to have an automated way of comparing classifiers and find the best ones. Considering that in time the training data will grow larger, or we'll include data from different domains, it is important to be able to quickly find the best solution for the particular domain.
As before, below is the diagram of the component followed by explanation of its sub-components :



The classifier will receive the training and test set. The test set will only be used for the evaluation. Each of the included classifier will be trained with the train set.
The following classifiers have been selected for testing :

1. **NaiveBayes Classifier**
2. **Maximum Entropy Classifier**
3. **CRF Classifier**
4. **HMM Classifier**

5. Neural Network Classifier

For these classifiers, we'll use a number of feature selection methods. In the diagram there is a feature selection subcomponent for each of the classifier, considering that the performance of each of them depends on the final subset of features and their transformation.

Bellow is a list of feature selection techniques that we are going to experiment with :

- **Mutual Information** : One of the most common feature selection methods is the Mutual Information of term t in class c. (Manning et al, 2008). This measures how much information the presence or absence of a particular term contributes to making the correct classification decision on c.
- **Chi square** : Another common feature selection method is the Chi Square. In feature selection we use it whether the occurrence of a specific term and the occurrence of a specific class are independent.
- **Wrapper feature selection strategy** : The goal here is to construct and select subsets of features that are useful to build an accurate classifier. This contrasts with the first 2 methods where the goal is finding/ranking all potentially relevant variables.
- **Feature selection using Decision Trees :**

One approach described in (*Chotirat "Ann" Ratanamahatana&Dimitrios Gunopulos, Scaling up the Naive Bayesian Classifier: Using Decision Trees for Feature Selection*) is to use the C4.5 algorithm in order to extract the most relevant features. The approach is described below :

1. Shuffle the training data and take 10% sample
2. Run C4.5 on data from step 1
3. Select a set of attributes that appear in the first 3 levels of the simplified decision tree as relevant features
4. Repeat 5 times steps 1-3
5. Form a union of all the attributes from the 5 rounds
6. Run the NaiveBayes classifier on the training data using only the final features selected in step 5.

The evaluator will store the results of classifiers for different feature selection strategies and will evaluate those results against the test set together with blended results like the average from all classifiers. The evaluator will suggest the best classifier together with the best feature selection strategy.

4 Bibliography

- NLTK.org/book (Natural Language Processing with Python by Steven Bird, Ewan Klein and Eduard Loper)
- http://www.sciencedirect.com/science/article/pii/S1532046409000033 (Feature selection techniques for maximum entropy based biomedical named entity recognition)
- http://stats.stackexchange.com/questions/104651/best-feature-selection-method-for-naive-bayes-classification
- http://blog.datumbox.com/using-feature-selection-methods-in-text-classification/
-