Entity Extraction

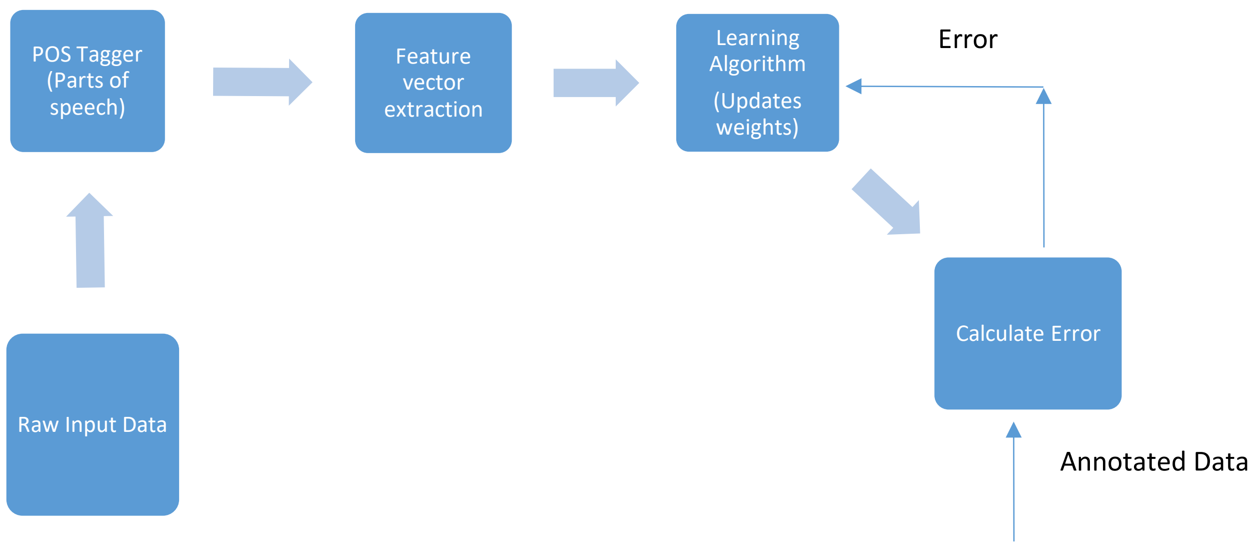
After reading through various papers I had made an algorithm which will combine the best features of Rule-based systems and machine learning based systems. The overall algorithm which I am trying to implement is a machine learning based algorithm.

Basic over view of the algorithm:

Input: Raw text data.

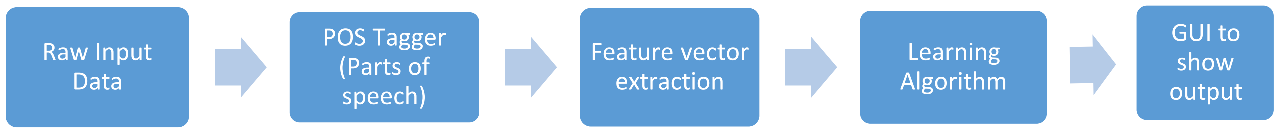
Output: Annotated text with annotations for Parts of Speech and Entities.

Flow chart:


POS Tagger (Parts of speech)
Feature vector extraction
Learning Algorithm (Updates weights)
Error
Calculate Error
Annotated Data
Raw Input Data

Learning Phase

Actual Phase



Algorithm:

- In first part of algorithm we will convert the data in any format into text data.
- In second part of algorithm we generated a semi annotated data from raw data. This semi annotated data will have only Parts of speech annotated.
- In third part we will generate the required Feature vectors from this semi annotated data which can be used in learning phase.
- In fourth part we will give these features to Learning algorithm in two modes.
- In learning mode learning algorithm will use these features to predict probable entities in the sentence. Then we validate the result with the known values of entities. The error in process is back propagated to adjust the weights in the learning algorithm.
- In Actual mode the predictions are given scores to reflect confidence of system. Based on the confidence values the predictions will be given as output or further processed.

PSEUDO CODE:

```
Step 1:  Load input data to Input_maker () module;

Step 2: Take its output as a file stream *Fp

Step 3: Pass *Fp = POS_Tagger (Fp) // this will create a semi annotated text in *Fp

Step 4: Create a vector of vectors which stores the features and take output of Feature_Extraction into it

        Vector<Vector <int>> features;

        features = Feature_Extraction (Fp);

Step 5: Pass features to learning algorithm and store results as Entities_Generated

Step 6: If (Mode == Training)

                Error = Compare (Entities_Generated, Actual_Entities)

                Weights = Update (Weights);

Step 7: else
```

```
        If (Confidence >= MIN_CONFIDENCE)
            Store results into OUTPUT vector
        else
            IMPROVE_COFIDENCE (features.This());
```

Step 8: Display the OUTPUT vector which will contain <Word, POS, Entity>

Implementation:

We are planning to implement the algorithm as 4 different modules which are as follows is as follows:

- Input_maker module
- POS_tagger module
- Learning_Algorithm module
- Output_maker module

Input_maker module:

This module handles the conversion of available data in any form into a text file.

POS_Tagger:

This module will take raw text data from Input module and tags all Parts of speech in it.

This module can be implemented as a rule based system or as a HMM based tagging module.

At present we are implementing this module as a HMM based module.

Unlike traditional methods this semi annotation of text will help learning algorithm to learn the structure of sentences.

Feature_Extraction:

This module will take semi annotated data and create feature vectors as required by learning algorithm.

Features we propose for this algorithm are:

(Here 2*N is the size of window considered)

1. Current word
2. Current word position in sentence
3. Current word's annotation and Capitalization
4. Prefixes and suffixes of current word
5. Size of current word

6. Special characters in current word
7. Numbers in the word
8. Previous N words
9. Next N words
10. Predictions of previous N words
11. Special characters in considered window with location
12. Prefixes and suffixes in current window with location
13. Annotations in window considered
14. Previous mentions of current word in text and its window.
15. Previous mentions prediction
16. Dictionary of current word (possible synonyms and antonyms)

Learning_Algorithm:

This module will take feature vector and returns the predicted entities in the sentence along with confidence value for each prediction.

This module can be implemented as a complete rule based one or as a learning algorithm. We will implement this module using Neural Networks.

If the confidence of the prediction is less then we can improve the accuracy by going for new techniques like using External Knowledge.

HMM based implementation of this is also possible.

Output_maker module:

This module of code will take the annotated text from the learning algorithm and display it in user friendly format.