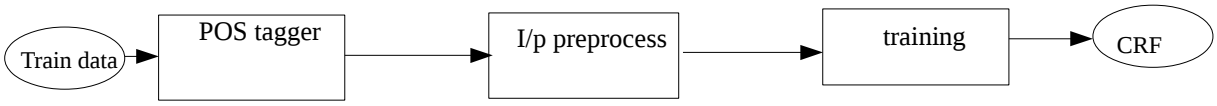
Design and implementation

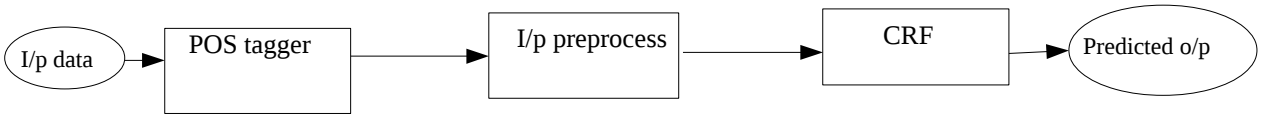
The named entity recognizer is made using MALLET in Python using scikit-learn toolkit.

The training and testing algorithm were designed as follows-

Training



Testing



Description of the sections

POS tagger

The Stanford POS tagger is used for the purpose of tokenizing the text and generating POS tags. An intermediate file was produced at this step.

I/p preprocess

The main steps of process are -

- Using the intermediate file from previous step to get the POS features.

- Checked capitalized words to include capitalization features

- Checked for STOPWORDS and included as features (removed afterwards)

-Checked for previous words and included as features

-Checked for previous POS tags and included as features (removed afterwards)

Training

The pre-processed text is readied for proper input for usage with the sequence labelling.

For this purpose, the MALLET sequence tagger is used which used a CRF as a model

The classifier is trained on 60% of the training data given using 100 iterations on the CRF. Initial training and testing acuuracy came out to be 0.8522 and 0.8222 respectively. Initial Fscore was just 0.0988 . This latter however, improved to 0.7069 with addition of features.

CRF

At training stage, the trained CRF model was outputted to a file which was used for tagging in testing phase.

For testing phase, the same file is given as an input for sequence tagging and the results computed.

Output processing(not shown above)

This step is just to make output in correct format as expected.

Evaluation

The model was trained as mentioned above and the results were computed against the the training data (without the labels). Then, for Fscore measurement it was compared against the training data (gold-output). An initial Fscore of 0.0955 was coming with a precision of 0.70 and recall 0.05

Study

Clearly, the initial recall over baseline was too low. On increasing the no. Of iterations, the recall percentage was too low. Increasing the no of the iterations in training, the Fscore went up to ~ 0.223.

Later on , following features in word are made and Fscore observed-

i. Baseline + Capitalization – 0.4067

ii. Baseline + Capitalization + POS tags - 0.5640

iii. Baseline + StopWords + Capitalization – 0.378

iv. Baseline + StopWords + Capitalization + POS tags – 0.542

v. Baseline + Capitalization + POS tags + previous words(w-1) – 0.7069

vi. Baseline + Capitalization + POS tags +previous POS tags + previous words(w-1) - 0.689

From above observation features with best Fscore was chosen i.e. - (v).

References/ Acknowledgement

1. python docs and stackoverflow for implementation
2. MALLET toolkit documentation for modelling the CRF.
3. Stanford POS tagger
4. Speech and Language processing - Dan Jurafsky and Martin