CS839 Project Stage-1

Group Number: **23**
Shang-Yen Yeh(syeh6@wisc.edu)
Chao Hsiang Chung(cchung49@wisc.edu)
Junxia Zhu(jzhu334@wisc.edu)

Environment

Please run our code under python 2.7 version. Sorry about the inconvenience.

Introduction

Entity Type:

We choose to extract person names from the documents. In our text files, we marked the entities with <person> and </person> tags.

For example, <person>Ortega</person> sings a story that…
<person>Giovanni Boccaccio</person> was an Italian writer…

Dataset Source:

Our 300 documents are mainly from Wikipedia, ABC News and The Guardian.

Dataset

Total number of mentions: 1937

set I:

number of documents: 200, number of mentions: 1348

set J:

number of documents: 100, number of mentions: 589

Attributes

We finally pick 26 attributes to train our model

- Check if input n-gram ends with ’s.
- Check if input n-gram contains country names.
- Check if input n-gram contains conjunction.
- Check if all the words of the input n-gram start with upper case.
- Check if the word in front of the input n-gram is a prefix, such as Mr., President, Secretary.
- Check if input n-gram is followed by common verbs, such as said, says, give.
- Check if input n-gram contains common verbs.

- Check if input n-gram contains prefix.
- Check if the word in front of the input n-gram is a preposition, such as by, with.
- Check if input n-gram contains organization names, such as committee , government, department.
- Check if input n-gram contains common human names, such as James, Jack, John.
- Check if input n-gram has any duplicate words.
- Count the word's occurrences in the article(only for single words). Return word's occurrences.
- Check if input n-gram has less than 2 lower case at each starting letter.
- Check if input n-gram contains month, such as January, February, December.
- Check if input n-gram contain words start or end with dash.
- Check if all the letters of input n-gram are capitalized.
- Count the length of input n-gram. Return length.
- Check if input n-gram is the start of a sentence.
- Check if the symbol after the input n-gram is comma.
- Check if the word after the input n-gram is ‘who’.
- Check if input n-gram contains comma.
- Check if input n-gram contains ‘who’.
- Check if input n-gram contains a suffix.
- Check if input n-gram starts with a suffix.
- Check if input n-gram contains exactly one dash.
- Check if input n-gram contains common adjective.

Model training

We do 10-fold cross validation and hyperparameter tuning on the following 5 classifiers.

- Decision Tree
- Support Vector Machine
- Naive Bayes
- Random Forest
- Logistic Regression

Classifier M

After performing cross validation and hyperparameter tuning on set I the first time, **Random Forest** performs best on set I. Performance of all 5 classifiers are listed as follows.

Classifier	Precision	Recall	F1
Logistic Regression	0.49	0.49	0.49
Decision Tree	0.61	0.17	0.27
SVM	0.61	0.17	0.27
Naive Bayes	0.49	0.49	0.49
Random Forest	0.49	0.60	0.54

Classifier X

After adding a few more attributes, performing cross validation and hyperparameter tuning on set I. **Random Forest** still performs best.

Classifier	Precision	Recall	F1
Logistic Regression	0.70	0.60	0.65
Decision Tree	0.86	0.57	0.69
SVM	0.81	0.59	0.68
Naive Bayes	0.30	0.93	0.45
Random Forest	0.87	0.64	0.74

We tried many different times to adjust our attributes. But the precision is still a little bit lower than requested. We plan to do post-processing, which is taking out overlapped n-grams(this part will be introduced detailed in next section). We believe this action will improve both our precision and recall to some degree. So we stop trying and apply our classifier X on set J to see whether we can reach 0.9 precision on unseen test set J after post-processing.

Post-processing

- **Take out overlapped n-grams**

  If our machine learning model predict an input n-gram as 1(a name), then we will traverse the document it belongs to and then take out those n-grams that are its subset. This will help improve our precision because partial names are excluded.

  For example, if 'Donald Trump' is regarded as a name.(This full name appears at document 027.txt, position 52-53.) Then we will take out 'Donald'(027.txt, position 52) and 'Trump'(027.txt, position 53) separately. This will only remove these two partial name so the same partial name appears in another position or in another document will not be influenced.

- **Reset some predicted labels**

  In attributes building part, we set several attributes that will definitely not be a name, such as contains a country name and contain an organization name. But unfortunately when predicting our machine learning algorithm still regard some n-grams with those attributes as 1(a name). So we reset the predicted label as 0 at post-processing step.

  For example, if 'York University' is regarded as a name, because our organization list contains 'University', so the predicted label of this n-gram will be set to 0.

You can check our post-processing code in **postProcessing.py**

Final Classifier Y

After training different machine learning models and performing rule-based post-processing, we finally got result on **Set J** as follows.

Classifier	Precision	Recall	F1
Logistic Regression	0.78	0.65	0.71
Decision Tree	0.93	0.56	0.70
SVM	0.93	0.61	0.74
Naive Bayes	0.40	0.97	0.56
Random Forest	0.91	0.65	0.76

Conclusion

Overall Random Forest performs best. Eventually we get 0.91 precision and 0.65 recall on set J.