Team Koala Ninja (10)
Ankit Vij
Amanpreet Singh Saini
Joel Haynie

Class: CS 839 - Data Science Spring 2018
Prof: Dr. AnHai Doan
Date: 03/03/2018
RE: Project Stage 1

Entity Type

Our team did the entity type of **"Person Names"**
Examples:
`<person>Donald Trump</>` `<person>Trump</>` `<person>Donald Trump Jr.</>`

Dataset

We have marked up 90 total files with 1777 mentions. The Training Set (i) is made up of 64 Files with 1172 Mentions. The Test Set (j) is made up of 26 Files with 605 Mentions.

Design of the pipeline

The design of the pipeline is as follows:

A. Data Processing & Cleaning

- **Whitelist**: We don't have any whitelist in place.
- **Pruning Rules**: Some of the candidates don't even make it to the labeled dataset either as a positive or as a negative example. Those which get filtered out have the following properties:
  - Generic stop words in the english language. Eg: about, any, but, was, etc.
  - Each word in the candidate string doesn't start with a capital letter.
  - Any candidate string which contains any letter except A-Z, a-z, whitespaces or a period.
  - Candidate strings above a length of 5 words (We decided on the limit of our name detection to be 5)
- **Black List**: We don't have a black list of names to strike out.
- **Pre-processing**: We are not modifying the candidate name in any manner but before getting to the actual meaningful feature extraction part of the pipeline, we are generating the below things based on the candidate names in the files:
  - An identifier based on uuid4
  - The unchanged candidate name itself
  - The true beginning position of the candidate name in the file as if the tags were not present
  - The true end position of the candidate name in the file as if the tags were not present
  - Frequency of the candidate name in the file
  - The label assigned to the candidate name based on the markups done

B. Features Selection:

- **Some letters capitalized -** This feature checks whether the name has one more than 2 capital letters. Helpful in cases where the names have three or more words.
- **At least one letter capitalized -** This feature checks whether the name has at least one capital letter. Helpful to weed out words with no capital letters.
- **First letter capitalized -** This feature checks whether the first letter of the name is capitalized or not.

- **Has suffix salutation -** This feature checks whether the name has suffix salutations. We have chosen the following salutations - “Jr.”, “Sr.”, “I”, “II”, “III”, “IV”
- **Position in the file -** The features provides the position of the name in the file.
- **Distance to closest period -** The feature provides the distance to the closet period from the name.
- **Distance to closest keywords -** This feature provides the distance to the closest keywords that could be helpful in identifying the names. The keywords that we have chosen are - "said", "Dr.", "Ph.D.", "PE", "president", "CEO", "by", "sen.", "senate", "officer", "attorney", "quoted", "spokeswoman", "spokesman", "reporter", "actress", "father", "son", "mother", "daughter", "Sheriff", "Justice", and "chief"
- **Frequency -** This feature specifies how many times the given name was found the document.
- **Contains period -** This feature specifies whether a name contains a period or not.
- **Contains keywords -** This feature specifies whether a name contains keywords that could be helpful in identifying the names. The keywords that we have chosen are - "said", "Dr.", "Ph.D.", "PE", "president", "CEO", "by", "sen.", "senate", "officer", "attorney", "quoted", "spokeswoman", "spokesman", "reporter", "actress", "father", "son", "mother", "daughter", "Sheriff", "Justice", and "chief"
- **Length -** This feature specifies the length of the name.
- **Number of capital letters** - This feature specifies the number of capital letters present in the name.
- **Distance to closest eol -** The feature provides the distance to the closet eol character from the name.

C. Machine Learning & Classifiers:

Implemented required classifiers:

1. Decision Tree
2. Random Forest
3. Support Vector Machine
4. Linear Regression
5. Logistic Regression

Post Processing: None

Additional ML: Ensemble Classification

Useful debug: Added prints for prediction hits and misses with false positive and false negative breakdown. Further dumping out the events that caused the miss for ease of debugging.

Results

We FIRST chose the Random Forest Classifier because it had the best Precision, Recall & F1 given below:
All below tests were averaged prediction of a set j, 10 folded.

Decision Tree Classifier:

	precision	recall	f1-score	support
0.0	0.9916	0.9945	0.9930	2583.6
1.0	0.8497	0.7894	0.8176	103.0
avg / total	0.9862	0.9866	0.9863	2686.6

Random Forest Classifier:

	precision	recall	f1-score	support
0.0	0.9902	0.9969	0.9936	2583.6
1.0	0.9062	0.7547	0.8223	103.0
avg / total	0.9870	0.9876	0.9870	2686.6

Support Vector Machine Classifier:

Linear Regression Classifier:

	precision	recall	f1-score	support
0.0	0.9616	0.9997	0.9803	2583.6
1.0	0.0000	0.0000	0.0000	103.0
avg / total	0.9248	0.9613	0.9427	2686.6

	precision	recall	f1-score	support
0.0	0.9653	1.0000	0.9823	2583.6
1.0	1.0000	0.0974	0.1765	103.0
avg / total	0.9666	0.9654	0.9514	2686.6

Logistic Regression Classifier:

	precision	recall	f1-score	support
0.0	0.9740	0.9995	0.9866	2583.6
1.0	0.9600	0.3303	0.4900	103.0
avg / total	0.9735	0.9738	0.9676	2686.6

We thought we were finished and ready do this write up! However, we discovered a bug in our file parsing that gave a HUGE hint to the classifiers, We re-ran our simulations with the “bug” fixed and obtained:

Decision Tree Classifier:

	precision	recall	f1-score	support
0.0	0.9915	0.9797	0.9856	20889
1.0	0.5075	0.7141	0.5933	612
avg / total	0.9777	0.9721	0.9744	21501

Random Forest Classifier:

	precision	recall	f1-score	support
0.0	0.9901	0.9915	0.9908	20889
1.0	**0.6954**	**0.6601**	**0.6773**	612
avg / total	0.9817	0.9821	0.9819	21501

Support Vector Machine Classifier:

	precision	recall	f1-score	support
0.0	0.9715	1.0000	0.9856	20889
1.0	0.0000	0.0000	0.0000	612
avg / total	0.9439	0.9715	0.9575	21501

Linear Regression Classifier:

	precision	recall	f1-score	support
0.0	0.9740	1.0000	0.9868	20889
1.0	1.0000	0.0899	0.1649	612
avg / total	0.9748	0.9741	0.9634	21501

Logistic Regression Classifier:

	precision	recall	f1-score	support
0.0	0.9788	0.9983	0.9885	20889
1.0	0.8214	0.2631	0.3985	612
avg / total	0.9744	0.9774	0.9717	21501

After refining our pruning rules, adding more features, and tweaking the models we eventually finalized on the Random Forest Classifier because it had the best precision, Recall & F1 after we were able to tweak it, as seen below:

Decision Tree Classifier:

	precision	recall	f1-score	support
0.0	0.8970	0.8803	0.8884	443.7
1.0	0.5874	0.6256	0.6050	120.3
avg / total	0.8310	0.8259	0.8280	564.0

Random Forest Classifier:

	precision	recall	f1-score	support
0.0	0.8861	0.9588	0.9210	443.7
1.0	**0.7831**	**0.5446**	**0.6413**	120.3
avg / total	0.8642	0.8704	0.8613	564.0

Support Vector Machine Classifier:

	precision	recall	f1-score	support
0.0	0.7892	1.0000	0.8822	443.7
1.0	0.8000	0.0148	0.0289	120.3
avg / total	0.7915	0.7899	0.7002	564.0

Linear Regression Classifier:

	precision	recall	f1-score	support
0.0	0.9196	0.7803	0.8441	443.7
1.0	0.4802	0.7477	0.5843	120.3
avg / total	0.8259	0.7734	0.7887	564.0

Logistic Regression Classifier:

	precision	recall	f1-score	support
0.0	0.8360	0.9543	0.8912	443.7
1.0	0.6499	0.3095	0.4183	120.3
avg / total	0.7963	0.8168	0.7903	564.0

As you can see we did NOT get to the targeted goal of **P = 0.9, R=0.6, and F1 = 0.72**.
We attained: **P = 0.78, R = 0.54. F1 = 0.64**.

We did not do any rule-based Post-processing. We however did do Ensemble learning to try to increase the accuracy of the Classifiers on the test set i. This code can be seen in the `\src\MLandMetrics\MLandMetrics.py`, more is discussed below and in the PipeLine section.

We ran our ensemble classifier, (10 k-fold on set j, size 40) on the Set i to see how degraded our performance:

```
Ensemble Learning:
             precision    recall  f1-score   support
        0.0     0.9166    0.7042    0.7965      2701
        1.0     0.3435    0.7073    0.4624       591
avg / total     0.8137    0.7047    0.7365      3292
hits: 2320 misses: 972 ( fp: 799 fn: 173 )
```

When run each Classifier individually we get the below results (SVM was removed):

```
Decision Tree Classifier:
             precision    recall  f1-score   support
        0.0     0.8788    0.8138    0.8451      2701
        1.0     0.3641    0.4873    0.4168       591
avg / total     0.7864    0.7552    0.7682      3292
hits: 2486 misses: 806 ( fp: 503 fn: 303 )
```

```
Random Forest Classifier:
             precision    recall  f1-score   support
        0.0     0.8686    0.9100    0.8888      2701
        1.0     0.4740    0.3706    0.4160       591
avg / total     0.7977    0.8132    0.8039      3292
hits: 2677 misses: 615 ( fp: 243 fn: 372 )
```

```
Logistic Regression Classifier:
             precision    recall  f1-score   support
        0.0     0.8478    0.9119    0.8787      2701
        1.0     0.3850    0.2521    0.3047       591
avg / total     0.7648    0.7934    0.7757      3292
hits: 2612 misses: 680 ( fp: 238 fn: 442 )
```

```
Linear Regression Classifier:
             precision    recall  f1-score   support
        0.0     0.9083    0.7149    0.8001      2701
        1.0     0.3396    0.6701    0.4508       591
avg / total     0.8062    0.7069    0.7374      3292
hits: 2327 misses: 965 ( fp: 770 fn: 195 )
```

We can see that performance was only degraded by about ~50% on data not trained on.

Future Improvements

We were not able to reach a precision of at least 90% and recall of at least 60%. We think that we could reduce the skew between the positives and negatives a bit more by adding more pruning rules. In the training set, we had ~1200 positives versus ~4400 negatives and in the test set, we had ~600 positives versus ~2600 negatives. We also did not employ any whitelist or blacklist, which can be used to increase the accuracy of our system further. In terms of features, we believe that we tried to cover all the cases, but we could certainly try using more features like finding the distance from the closest preposition and action verb, which could possibly improve the performance.