839 projecct 1 report

Zhendong Zhou Yunwen Jiang Yixin Chen

In this project, we plan to extract names from documents. All the documents are articles and novels from the internet. We tagged the names in each document with labels like :<name>…</name>. For examples:

My name is <name>Michael</name>
The old <name>Josh</name> is sleepy.

Among 300 documents, We have marked 1346 names.

In set I, there are 200 documents, and there are 904 mentions within them.

In set J, there are 100 documents, and there are 442 mentions within them.

All the negative variables are chosen to be started with capitalized letter, which will make it much harder. But we think that it's worth trying.

First, we decided to apply 17 features:

1. If there is honorific in front of the sample, like: Miss, Mr, professor,etc.
2. If all the characters are capitalized.
3. If there is comma before and after the word.
4. If there is "a", "an", "the" before or after the word.
5. If there is special adj. before the word like "poor" and "cute".
6. If there is special adj after the word.
7. If there is special word before the word like "sister" and "brother".
8. If there is some special word before the word like "said" and "named".
9. If it's the end of the sentence.
10. If it's ended up or started with some important words like "who".
11. If there is prep. before or after the word.
12. If the word before it is ended up with "ed".
13. If the word after it is ended up with "ed"
14. The length of the word.
15. If the word starts with capital letter.
16. If there is "'s" after the word.
17. If it's the start of the sentence.

Then we transferred the document into a data frame containing the features and fitted the machine learning algorithms on that.

The algorithms I applied are Logistic, Linear Regression, Decision Tree, Random Forest, SVM and MLP.

Finally we got some conclusions.

Classifier M is Decision tree. The output is shown below:

```
Decision Tree
              precision    recall  f1-score   support

    negative       0.86      0.96      0.90      3504
    positive       0.69      0.38      0.49       904
```

Obviously, it's not good enough. So we decide to debug it.

After taking a look at the samples that are not classified right, we found that it's not proper to add rules. For the words appear before and after the target are not gathering in some words. There are so many words only appear once or twice, wich may not provide us any insight.

But considering that we are only interested in the precision and recall of the positive samples, so we dedcide to change the parameter of the algorithm when making the decision.

Due to the high imbalance of our dataset, we did SMOTE on the both of the train and test dataset to make it more precise.

After debugging, the best classifier X is Random Forest. The output is shown below:

```
Random Forest
              precision    recall  f1-score   support

    negative       0.70      0.92      0.79      3504
    positive       0.88      0.61      0.72      3504
```

Which seems to be much better, and that's what we want.

After attempting for couple of times, we find that we can't make it better, so we decide to try it on the Set J, the output is shown below:

```
Random Forest
              precision    recall  f1-score   support

    negative       0.71      0.93      0.81      1961
    positive       0.90      0.63      0.74      1961
```

One thing need to say is that, due to the randomness of the RandomForest, the output will change every time we run the code, so it's normal not succeeding the threshold in some situations.

Thanks god it’s better than that on the development set.

What we have learnt from this project is, the data itself, should always from the same sample. For example: we have documents from different kind of novels, so some animals may act like human like in fairy tale, etc. These situations are ones we can’t deal with. In the future, we need to pay more attention on the data itself before conducting the project.