Intermediate Report

Weikai Mao

1. Project description

As for the Course Project Proposal. I want to do an NLP project in Kaggle: Toxic Comment Classification Challenge (https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/overview). In this project, I need to create a model to predicts the probability of each type of toxicity for each comment. The dataset includes the Wikipedia comments which have been labeled for toxic behavior (toxic, severe_toxic, obscene, threat, insult, identity_hate).

2. Source and format of the data

The data is in csv format. There are 8 columns: `id` (string), `identity_hate` (integer), `toxic` (integer), `severe_toxi` (integer), `obscene` (integer), `threat` (integer), `insult` (integer), and `comment_text` (string). The feature `id` is useless, so I dropt it.

Apart from `id` and `comment_text` , the other 6 features indicate whether this comment text is labeled for the corresponding toxic behavior.

```
%pyspark

path = "/home/wkm/MEGAsync/Rutgers/2019Fall/Massive data storage and retrival/Project/jigsaw-toxi
# use pyspark to read the data as spark dataframe
train = spark.read.load(path+"train_for_spark.csv", format="csv", sep=",", inferSchema="true", he
```

```
%pyspark

train = train.drop('_c0','id') # drop the useless columns
train.fillna(0)
train.show() # show top 20 rows
```

```
+-------------+-----+------------+-------+------+------+--------------------+
|identity_hate|toxic|severe_toxic|obscene|threat|insult|        comment_text|
+-------------+-----+------------+-------+------+------+--------------------+
|            0|    0|           0|      0|     0|     0|Explanation Why t...|
|            0|    0|           0|      0|     0|     0|D'aww! He matches...|
|            0|    0|           0|      0|     0|     0|Hey man, I'm real...|
|            0|    0|           0|      0|     0|     0|""" More I can't ...|
|            0|    0|           0|      0|     0|     0|You, sir, are my ...|
|            0|    0|           0|      0|     0|     0|"""  Congratulati...|
|            0|    1|           1|      1|     0|     1|COCKSUCKER BEFORE...|
|            0|    0|           0|      0|     0|     0|Your vandalism to...|
|            0|    0|           0|      0|     0|     0|Sorry if the word...|
|            0|    0|           0|      0|     0|     0|alignment on this...|
|            0|    0|           0|      0|     0|     0|""" Fair use rati...|
```

```
|           0|    0|          0|      0|     0|     0|bbq   be a man an...|
|           0|    1|          0|      0|     0|     0|Hey... what is it...|
|           0|    0|          0|      0|     0|     0|Before you start ...|
```

```
%pyspark

train.printSchema() # schema of the spark dataframe
print("There are {} observations and {} features in training data set.".format(train.count(), len
```

```
root
 |-- identity_hate: integer (nullable = true)
 |-- toxic: integer (nullable = true)
 |-- severe_toxic: integer (nullable = true)
 |-- obscene: integer (nullable = true)
 |-- threat: integer (nullable = true)
 |-- insult: integer (nullable = true)
 |-- comment_text: string (nullable = true)

There are 159571 observations and 7 features in training data set.
```

3. Data analysis

```
%pyspark

# descriptive statistics
train.describe().show()
```

```
+-------+------------------+------------------+------------------+------------------+-----------------+-------------------+------------------
-+--------------------+
|summary|      identity_hate|             toxic|       severe_toxic|          obscene|             threat|             insul
t|        comment_text|
+-------+------------------+------------------+------------------+------------------+-----------------+-------------------+------------------
-+--------------------+
|  count|            159571|            159571|            159571|            159571|           159571|             159571|             15957
1|              159571|
|   mean|0.008804858025581 09|0.09584448302009764|0.009995550569965721|0.052948217407925|0.002995531769557125|0.0493636061690407
4|                null|
| stddev|0.09342048594149767| 0.2943787715999705| 0.09947714085748408|0.223930832915411| 0.05464958623142267| 0.216626717276817
9|                null|
|    min|                 0|                 0|                 0|                0|                 0|
0|  Thank you. Now ...|
|    max|                 1|                 1|                 1|                1|                 1|
1|Sensual Pleasure...|
+-------+------------------+------------------+------------------+------------------+-----------------+-------------------+------------------
-+--------------------+
```

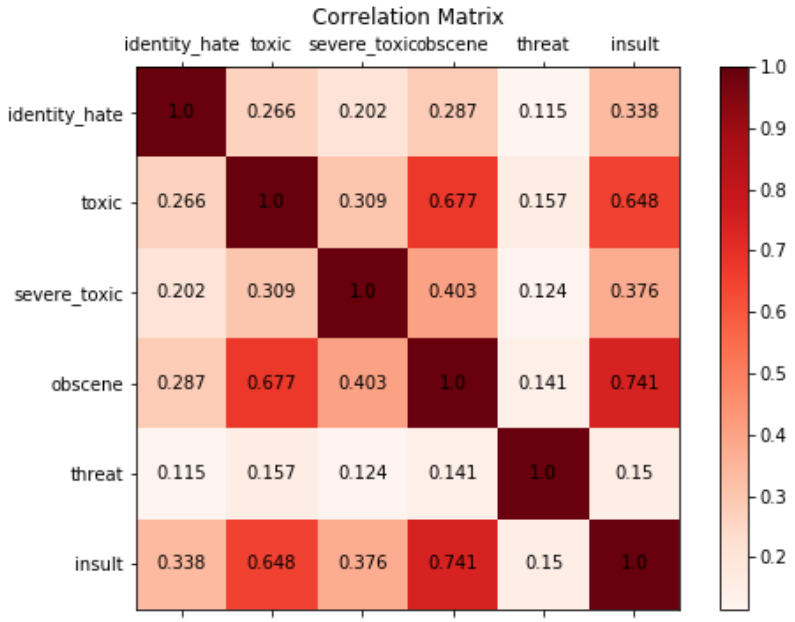
The table above shows that the frequency of toxic behaviour is highest, whereas the frequency of threat behaviour is lowest.

```
%pyspark

# calculate pair wise frequency
train.crosstab("obscene", "insult").show()
train.crosstab("toxic", "identity_hate").show()
```

```
+--------------+------+----+
|obscene_insult|     0|   1|
+--------------+------+----+
|             1|  2294|6155|
|             0|149400|1722|
+--------------+------+----+

+------------------+------+----+
|toxic_identity_hate|     0|   1|
+------------------+------+----+
|                 1| 13992|1302|
|                 0|144174| 103|
+------------------+------+----+
```

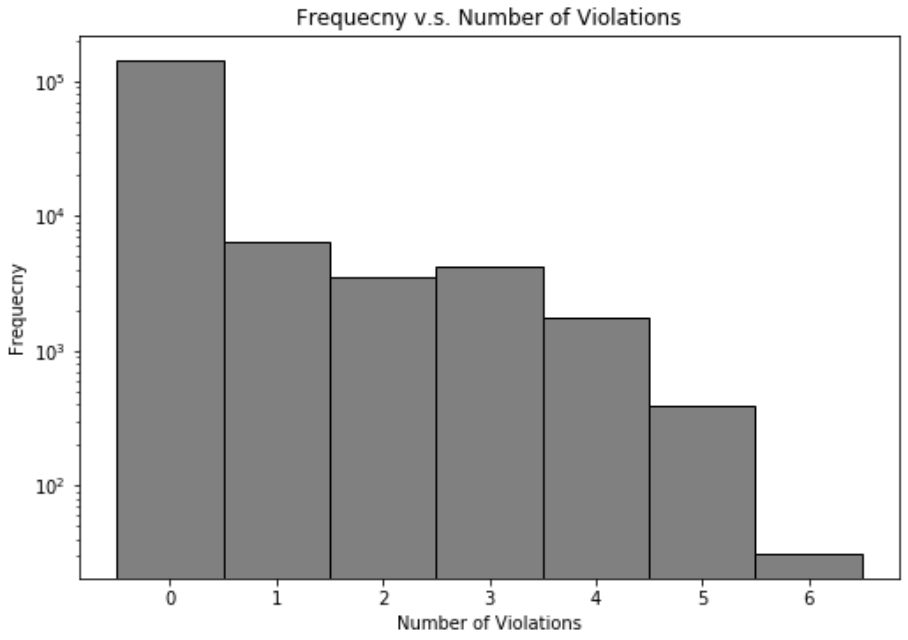
```
%pyspark

# compute correlation matrix
from pyspark.mllib.stat import Statistics
import pandas as pd
import matplotlib.pyplot as plt

features = train.rdd.map(lambda row: row[:-1])
features.persist()

corr_mat = Statistics.corr(features, method="pearson")

corr_df = pd.DataFrame(corr_mat)
corr_df.index, corr_df.columns = train.columns[:-1], train.columns[:-1]
print(corr_df.to_string())


fig, ax = plt.subplots()
im = ax.imshow(corr_df)
im = ax.matshow(corr_df, cmap=plt.cm.Reds)
# Loop over data dimensions and create text annotations.
for i in range(len(corr_df)):
    for j in range(len(corr_df)):
        text = ax.text(j, i, round(corr_df.iloc[i, j], 3), \
                       ha="center", va="center")

#labels
ax.set_xticklabels([""]+train.columns[:-1], minor=False)
ax.set_yticklabels([""]+train.columns[:-1], minor=False)

ax.figure.colorbar(im, ax=ax)
ax.set_title("Correlation Matrix")
plt.show()
```

```
              identity_hate     toxic  severe_toxic   obscene    threat    insult
identity_hate       1.000000  0.266009      0.201600  0.286867  0.115128  0.337736
toxic               0.266009  1.000000      0.308619  0.676515  0.157058  0.647518
severe_toxic        0.201600  0.308619      1.000000  0.403014  0.123601  0.375807
obscene             0.286867  0.676515      0.403014  1.000000  0.141179  0.741272
threat              0.115128  0.157058      0.123601  0.141179  1.000000  0.150022
insult              0.337736  0.647518      0.375807  0.741272  0.150022  1.000000
```



From the correlation matrix above, we can see that either two of `toxic` , `obscene` and `insult` have high correlation.

```
%pyspark

# histogram
import matplotlib.pyplot as plt

sumi = features.map(lambda x: sum(x))
plt.hist(sumi.collect(), 7, range=[-0.5, 6.5], facecolor="grey", alpha=1, histtype='bar', ec='bla
plt.yscale("log")
plt.ylabel('Frequecny')
plt.xlabel('Number of Violations')
plt.title('Frequecny v.s. Number of Violations')

plt.show()
```



Took 3 sec. Last updated by anonymous at October 22 2019, 8:20:14 PM. (outdated)

The histogram above shows that most of comments has not been labeled for toxic behaviours. Very few of comments has been labeled for all 6 toxic behaviours.

4. Preprocessing comments

4.1 Word segmentation

```
%pyspark

textRdd = train.rdd.map(lambda column: column[-1])
textRdd.persist()
textRdd.take(1)
```

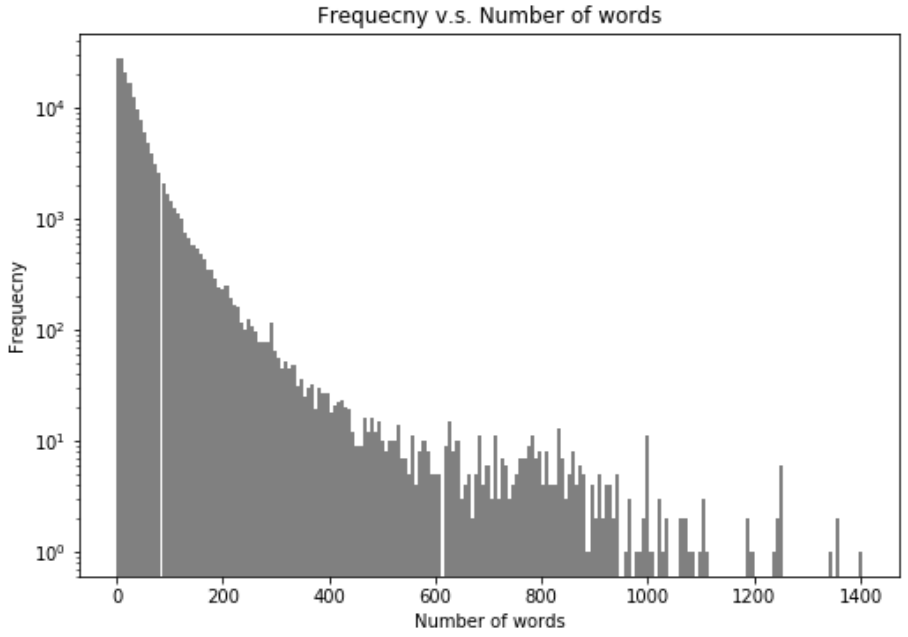
```
["Explanation Why the edits made under my username Hardcore Metallica Fan were reverted? They weren't vandalisms, just closure on some GAs after I voted at New York Dolls FAC. And please don't remove the template from the talk page since I'm retired now.89.205.38.27"]
```

```
%pyspark

from pyspark.ml.feature import RegexTokenizer

# sentence tokenizer
tokenizer = RegexTokenizer(inputCol="comment_text", outputCol="words", pattern="\\W")
wordsDf0 = tokenizer.transform(train)
wordsRdd0 = wordsDf0.rdd.map(lambda column: column[-1])
wordsRdd0.take(2)
```

```
[['explanation', 'why', 'the', 'edits', 'made', 'under', 'my', 'username', 'hardcore', 'metallica', 'fan', 'were', 'reverted', 'they', 'weren', 't', 'vandalisms', 'just', 'closure', 'on', 'some', 'gas', 'after', 'i', 'voted', 'at', 'new', 'york', 'dolls', 'fac', 'and', 'please', 'don', 't', 'remove', 'the', 'template', 'from', 'the', 'talk', 'page', 'since', 'i', 'm', 'retired', 'no
```

```
w', '89', '205', '38', '27'], ['d', 'aww', 'he', 'matches', 'this', 'background', 'colour', 'i', 'm', 'seemingly', 'stuck', 'wit
h', 'thanks', 'talk', '21', '51', 'january', '11', '2016', 'utc']]
```

```
%pyspark

import matplotlib.pyplot as plt

leni0 = wordsRdd0.map(lambda x: len(x))

plt.hist(leni0.collect(), 200, facecolor="grey", alpha=1, histtype='bar')
plt.yscale("log")
plt.ylabel('Frequecny')
plt.xlabel('Number of words')
plt.title('Frequecny v.s. Number of words')

plt.show()
```



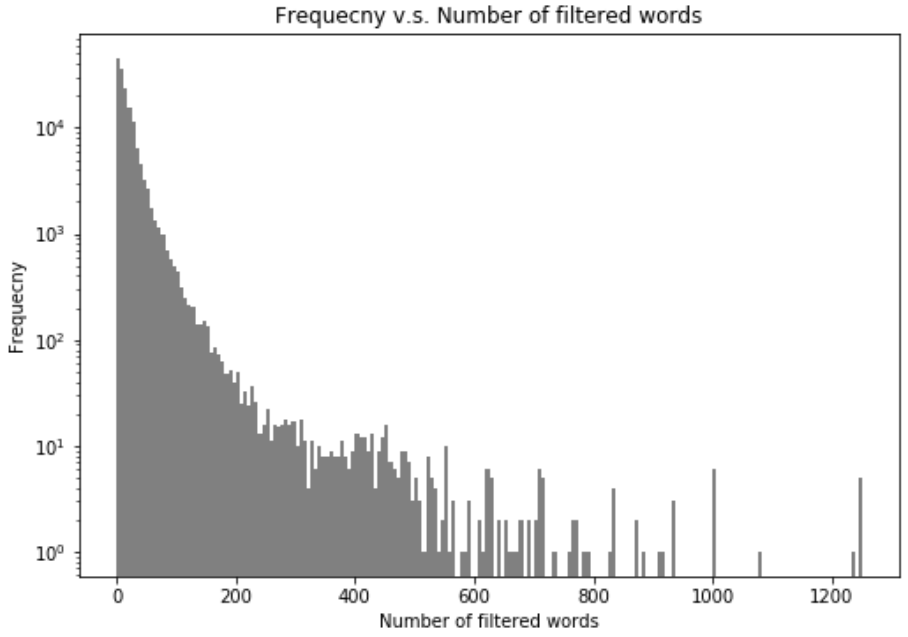
4.2 Removing stop words

```
%pyspark

from pyspark.ml.feature import StopWordsRemover

remover = StopWordsRemover(inputCol="words", outputCol="filtered")
wordsDf1 = remover.transform(wordsDf0)
wordsRdd1 = wordsDf1.rdd.map(lambda column: column[-1])
wordsRdd1.take(2)
```

```
[['explanation', 'edits', 'made', 'username', 'hardcore', 'metallica', 'fan', 'reverted', 'weren', 'vandalisms', 'closure', 'ga
s', 'voted', 'new', 'york', 'dolls', 'fac', 'please', 'remove', 'template', 'talk', 'page', 'since', 'm', 'retired', '89', '205',
'38', '27'], ['d', 'aww', 'matches', 'background', 'colour', 'm', 'seemingly', 'stuck', 'thanks', 'talk', '21', '51', 'january',
'11', '2016', 'utc']]
```

```
%pyspark
leni1 = wordsRdd1.map(lambda x: len(x))

plt.hist(leni1.collect(), 200, facecolor="grey", alpha=1, histtype='bar')
plt.yscale("log")
plt.ylabel('Frequecny')
plt.xlabel('Number of filtered words')
plt.title('Frequecny v.s. Number of filtered words')
```

```
Text(0.5, 1, 'Frequecny v.s. Number of filtered words')
```




```
%pyspark

wordsDf2 = wordsDf1.drop("comment_text", "words")
wordsDf2.show()
wordsDf2.toPandas().to_csv(path+"train_processed.csv")
```

```
+-------------+-----+------------+-------+------+------+--------------------+
|identity_hate|toxic|severe_toxic|obscene|threat|insult|            filtered|
+-------------+-----+------------+-------+------+------+--------------------+
|            0|    0|           0|      0|     0|     0|[explanation, edi...|
|            0|    0|           0|      0|     0|     0|[d, aww, matches,...|
|            0|    0|           0|      0|     0|     0|[hey, man, m, rea...|
|            0|    0|           0|      0|     0|     0|[make, real, sugg...|
|            0|    0|           0|      0|     0|     0|[sir, hero, chanc...|
|            0|    0|           0|      0|     0|     0|[congratulations,...|
|            0|    1|           1|      1|     0|     1|[cocksucker, piss...|
|            0|    0|           0|      0|     0|     0|[vandalism, matt,...|
|            0|    0|           0|      0|     0|     0|[sorry, word, non...|
|            0|    0|           0|      0|     0|     0|[alignment, subje...|
|            0|    0|           0|      0|     0|     0|[fair, use, ratio...|
|            0|    0|           0|      0|     0|     0|[bbq, man, lets, ...|
|            0|    1|           0|      0|     0|     0|[hey, talk, exclu...|
|            0|    0|           0|      0|     0|     0|[start, throwing,...|
|            0|    0|           0|      0|     0|     0|[oh, girl, starte...|
```

5. Ideas for the next phase

This data set can be used to build a classification model. In the next phase, I will do word embedding and modeling by neural network.