Text Classification on Hate Speech

2nd Semester Data Science Master
Beuth University of Applied Sciences Berlin

by Arndt, Ana, Christian, Ervin, Malte

Content

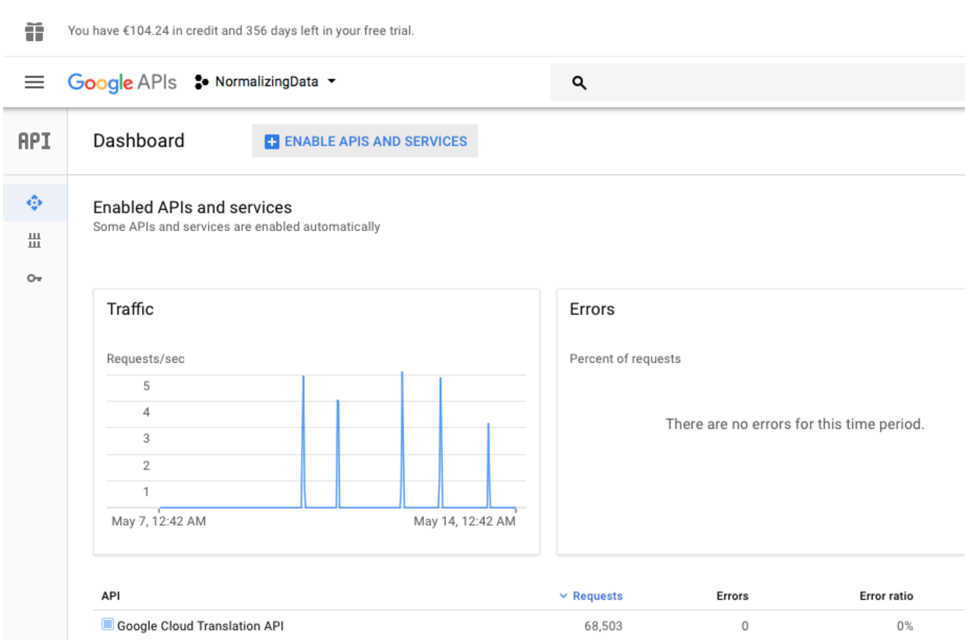
1. Preprocessing
2. Baseline
3. Improvements
4. Ensembles
5. Recap

Data Preprocessing and Normalization

Improve the performance of the model applying some simple pre-processing

- Translation into English
- Only ASCII characters (unidecode)
- Remove special characters
- Change Emojis to words

Google Translation API Requests



Replace Characters

```
def replacetext(text):
    for key, value in REPLACE_TO.items():
        text = text.replace(key, value)
    return text

REPLACE_TO = {
   ':)': 'happy',':(': 'sad',':P': 'funny','@': 'at','&': 'and','i\'m': 'i am'
,'don\'t': 'do not','can\'t': 'can not',
   '.': ' ',',': ' ',':': ' ', ';': ' ','!': ' ','\'': ' ','?': ' ', '(': ' ',
')': ' ', '[': ' ', ']': ' ', '-': ' ',
   '#': ' ', '=': ' ', '+': ' ', '/': ' ', '"': ' ', '0': ' zero ', '1': ' one
', '2': ' two ', '3': ' three ', '3': ' three ',
   '4': ' four ','5': ' five ', '6': ' six ', '7': ' seven ', '8': ' eight ',
'9': ' nine ',
   '|':' ', '$':' ', '%':' ', '^':' ', '*':' ', '_':' ', '{':' ', '}':' ', '<'
:' ', '>':' ',
   '\n': ' '
}
```

Majority class classifier

In [29]:
```
score_preds(y_test, np.zeros(y_test.shape)) #set all predictions to non-toxic (
0)
```

```
confusion matrix:
[[57888     0]
 [ 6090     0]]
classification report:
             precision    recall  f1-score   support

          0       0.90      1.00      0.95     57888
          1       0.00      0.00      0.00      6090

avg / total       0.82      0.90      0.86     63978

f1 macro: 0.4750
f1 micro: 0.9048
```

- By only assigning all fitted values to the majority class we get a F1 score of 90%.
- This is, because the test dataset is imbalanced and contains only 10% toxic comments.

Single model - baseline

using skift (scikit fasttext) - scikit-learn wrappers for Python (GitHub (https://github.com/shaypal5/skift))

In [41]:

```
X_train, X_test, y_train, y_test = load_train_test_data("data/train.csv", "data/test.csv", "data/test_labels.csv", "toxic")

skift_clf = skift.FirstObjFtClassifier()
skift_clf.fit(X_train, y_train)
preds = skift_clf.predict(X_test)
score_preds(y_test, preds)
print("f1 micro on training data: %0.4f" % (skift_clf.score(X_train, y_train)))
```

```
confusion matrix:
[[54325  3563]
 [ 1218  4872]]
classification report:
             precision    recall  f1-score   support

          0       0.98      0.94      0.96     57888
          1       0.58      0.80      0.67      6090

avg / total       0.94      0.93      0.93     63978

f1 macro: 0.8143
f1 micro: 0.9253
f1 micro on training data: 0.9722
```

Single model - parameters

In [3]:

```
X_train, X_test, y_train, y_test = load_train_test_data("data/train_unidecode.c
sv", "data/test_unidecode.csv", "data/test_labels.csv", "toxic")

skift_clf = skift.FirstObjFtClassifier(wordNgrams=2, maxn=3, dim=300)
skift_clf.fit(X_train, y_train)
preds = skift_clf.predict(X_test)
score_preds(y_test, preds)
print("f1 micro on training data: %0.4f" % (skift_clf.score(X_train, y_train)))
```

```
confusion matrix:
[[55608  2280]
 [ 1939  4151]]
classification report:
             precision    recall  f1-score   support

          0       0.97      0.96      0.96     57888
          1       0.65      0.68      0.66      6090

avg / total       0.94      0.93      0.93     63978

f1 macro: 0.8132
f1 micro: 0.9341
f1 micro on training data: 0.9586
```

Check common errors - false negatives

What commonnalities have the false negatives? Check common errors ...

False negatives: labeled as toxic, not identified as toxic

- "well it **sucks** to have a university to be nicknameless and it s the first time in ncaa history that it has happened"
- " intolerance in india india is a generator of liars like you"
- "look you re a pedant and fetzer is a jew hater on press tv in the uk today september two nd two zero one one he said that the israelis were behind nine one one the man is a complete fool "
- "not even every sexual person fantasizes while masturbating most males do but many females do not i think most libidinous asexuals masturbate for the same reason they would scratch themselves if they were itchy "

- **few or no predominantly abusive-use words in a "normal speech context"**
- **many would actually not label these comments as toxic**
- **could argue that the classifier actually does a good job**

but:

- "hey **shithead** stop vandilizing articles "

Check common errors - false positives

What commonnalities have the false positives? Check common errors …

False Positives: labeled as non-toxic, identified as toxic

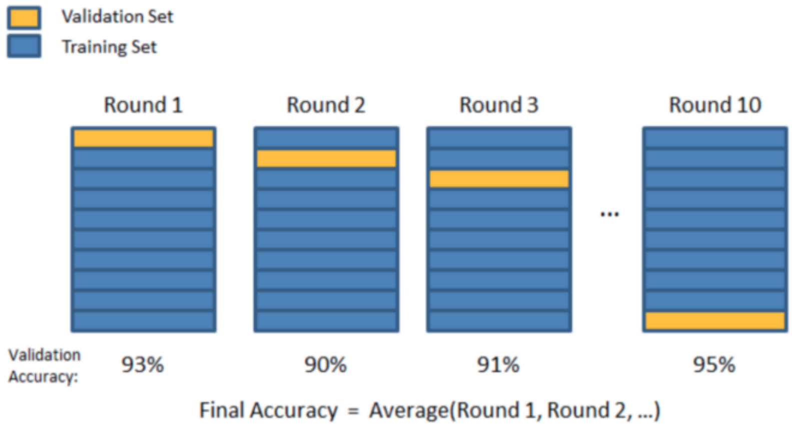
- "i will burn you to hell if you revoke my talk page access" - **wrongly labeled**
- " buffoon synonyms bozo buffo clown comedian comic fool harlequin humorist idiot jerk jester joker merry andrew mime mimic mummer playboy prankster ridicule stooge wag wit zany " - **special context**
- " gay he s gay too it should be noted that he has a male partner " **non-abusive use of a term that is predominantly used in an abusive way in the corpus**

Building fastText Ensembles

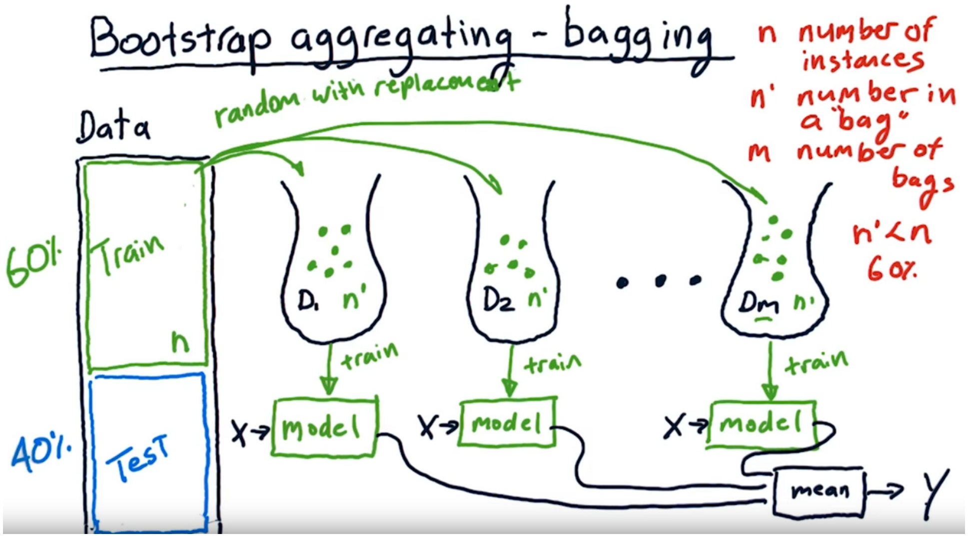
Make predictions with a collection of classifiers on a dataset X.

- K Fold
- Stratified K Fold
- Bagging
- Bagging with Oversampling

K Folds



Validation Set
Training Set
Round 1
Round 2
Round 3
Round 10
...
Validation Accuracy:
93%
90%
91%
95%
Final Accuracy = Average(Round 1, Round 2, ...)


Bootstrap aggregating - bagging
random with replacement
Data
60%
Train
n
40%
Test
D1 n'
D2 n'
Dm n'
train
train
train
X→ model
X→ model
X→ model
mean → Y
n number of instances
n' number in a "bag"
m number of bags
n'<n
60%

Result comparison - single models

Single model, original data, default parameters

```
[[54325  3563]
 [ 1218  4872]]
```

f1 macro: 0.8143
f1 micro: 0.9253
f1 micro on training data: 0.9722

Single model, preprocessed data, default parameters

```
[[54324  3564]
 [ 1222  4868]]
```

f1 macro: 0.8141
f1 micro: 0.9252
f1 micro on training data: 0.9722

Single model, preprocessed data, wordNgrams=2, maxn=3, dim=300

```
[[55608  2280]
 [ 1939  4151]]
```

f1 macro: 0.8132
f1 micro: 0.9341
f1 micro on training data: 0.9586

Result comparison - ensembles

10 Folds, preprocessed data, minn=3, maxn=3, wiki-news-300d-1M-subword.vec

```
[[54887  3001]                                      f1 macro: 0.8194
 [ 1431  4659]]                                     f1 micro: 0.9307
```

10 Stratified Folds, preprocessed data, minn=3, maxn=3, wiki-news-300d-1M-subword.vec

```
[[54915  2973]                                      f1 macro: 0.8200
 [ 1436  4654]]                                     f1 micro: 0.9311
```

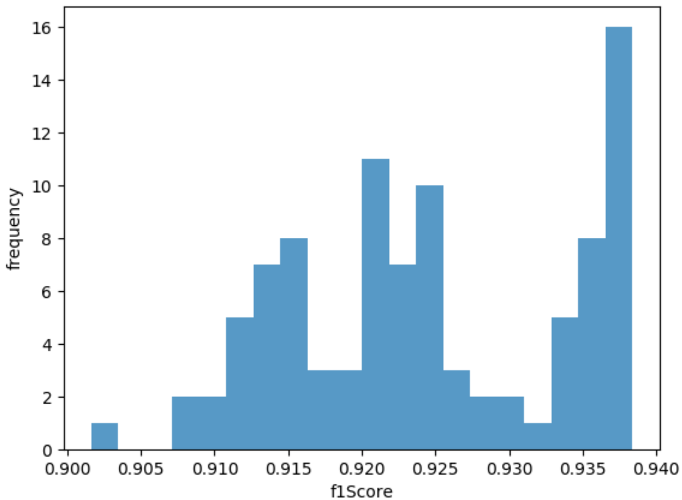
Oversampling ensemble, preprocessed data, 8 bags, lr=0.2

```
[[56181  1707]                                      f1 macro: 0.7987
 [ 2477  3613]]                                     f1 micro: 0.9346
```

Oversampling ensemble, preprocessed data, 96 bags, lr=0.2

```
[[54913  2975]                                      f1 macro: 0.8195
 [ 1444  4646]]                                     f1 micro: 0.9309
```

Oversampling Ensemble Histogram



Identity hate

In [8]:

```
X_train, X_test, y_train, y_test = load_train_test_data("data/train_unidecode.c
sv", "data/test_unidecode.csv", "data/test_labels.csv", "identity_hate")

skift_clf = skift.FirstObjFtClassifier()
skift_clf.fit(X_train, y_train)
preds = skift_clf.predict(X_test)
score_preds(y_test, preds)
print("f1 micro on training data: %0.4f" % (skift_clf.score(X_train, y_train)))
```

```
confusion matrix:
[[63026   240]
 [  479   233]]
classification report:
             precision    recall  f1-score   support

          0       0.99      1.00      0.99     63266
          1       0.49      0.33      0.39       712

avg / total       0.99      0.99      0.99     63978

f1 macro: 0.6938
f1 micro: 0.9888
f1 micro on training data: 0.9934
```

Conclusion

- Random results - due to initialization of neural net's weights - make result comparison difficult
- Ensembles to stabilize the results
- Really unclear on how some parameters improve the score i.e. pretrained vectors
- Usage within scikit-learn difficult, if you don't have numeric predictors

More ideas

- GridSearch on "good" fastText hyperparameters
- Generate many models and persist one that scores high
- Continousliy improve this persisted model
- More detailed comparison of the probabilities of FPs/FNs of different models

Tfidf Method using Scikit vectorizer

This method was inspired by one of the Kaggle Competitors who used sklearn to implement a Logistic regression with words & char n grams. And his work achieved a better score only to mention that it doesn't use fastText at all for it's implementation.

Source: https://www.kaggle.com/tunguz/logistic-regression-with-words-and-char-n-grams/code (https://www.kaggle.com/tunguz/logistic-regression-with-words-and-char-n-grams/code)

A few edits were made to create the following result:

- Test CV score (ROC AUC) for class toxic is 0.957
- Test CV score (ROC AUC) for class identity_hate is 0.975