TOXIC COMMENT CLASSIFIER

Shaunak Phatak

OUTLINE

- Problem Context
- Problem Definition
- Data Wrangling and Exploratory Data Analysis (EDA)
- Modeling Results
- Conclusion

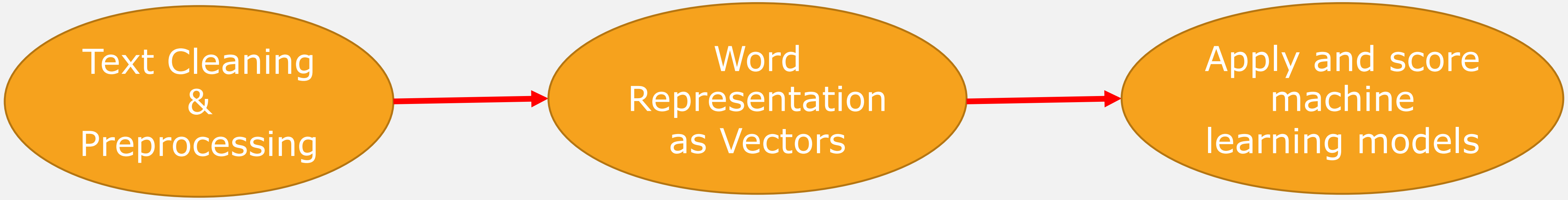
PROBLEM CONTEXT

Reference: https://disruptive.asia/critical-troubling-cyberbullying-statistics-emerge-in-2020/

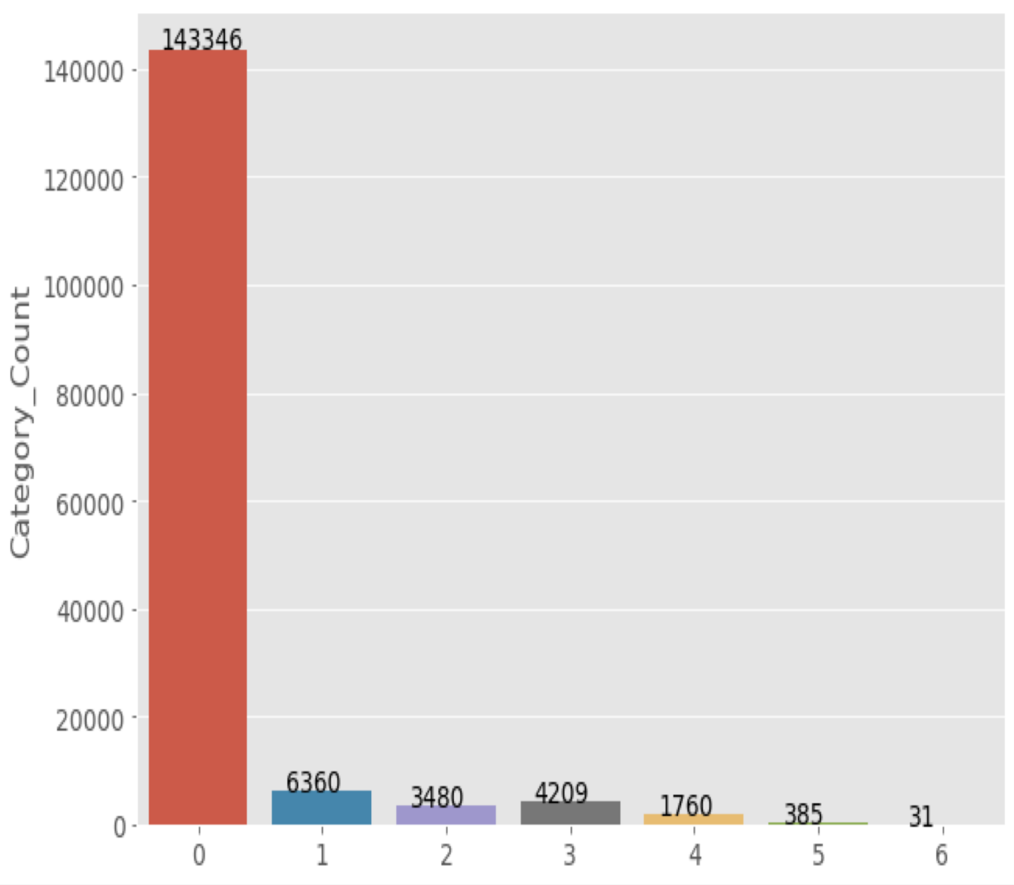
- Ease of global communication and debate due to social media platforms like Facebook
- Unfortunate downside: Easy for people to engage in online abuse and cyber bullying
- Manually flagging this behavior is not feasible due to the scale of data generated
- Natural Language Processing (NLP) can be utilized to automate such a task
- The goal of this project is to create a multi-label toxic comment classifier using a Wikipedia comments dataset from Kaggle
- Various NLP and machine learning techniques will be tested to create an optimal model

PROBLEM DEFINITION

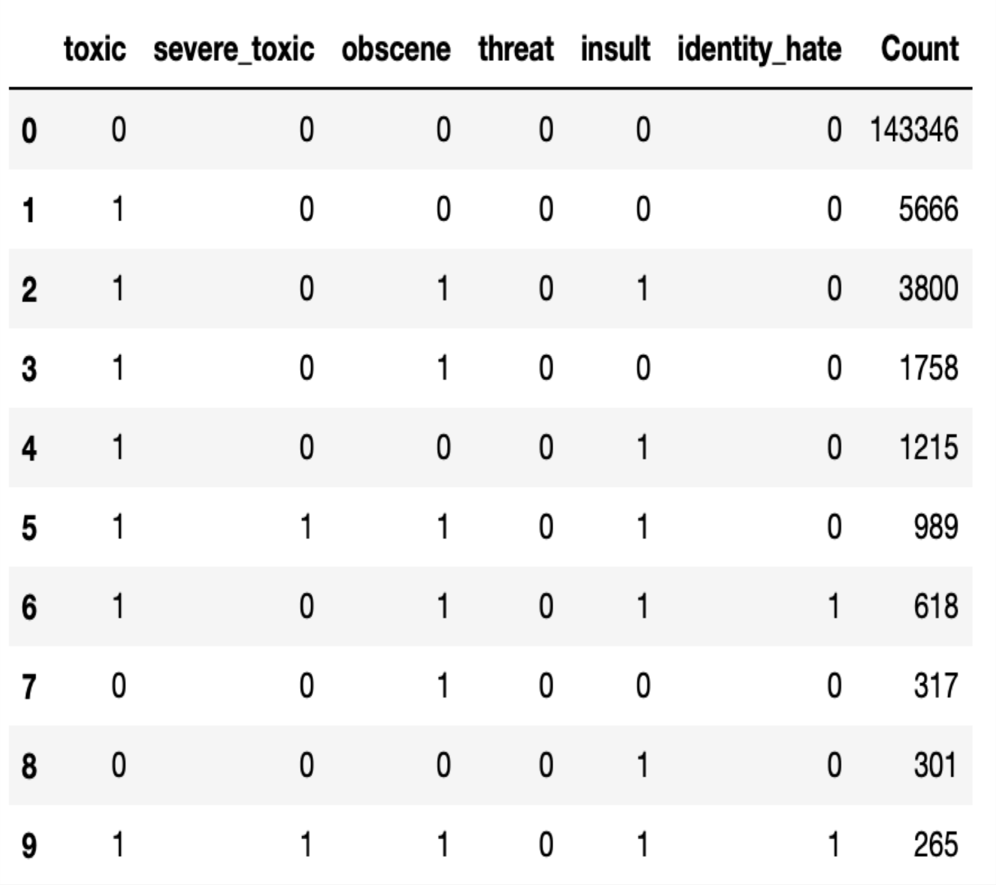
- Create a multi-label classifier to classify comments into 6 categories: toxic, severe toxic, threat, insult, obscene and identity hate
- Dataset Information:
  - Multi-label dataset obtained from a Kaggle competition
  - ~160,000 comments which can have multiple labels
- The project pipeline involves the following high-level work flow:


Text Cleaning & Preprocessing
Word Representation as Vectors
Apply and score machine learning models

DATA WRANGLING AND EDA



- Comment distribution by number of labels
- Highly imbalanced dataset



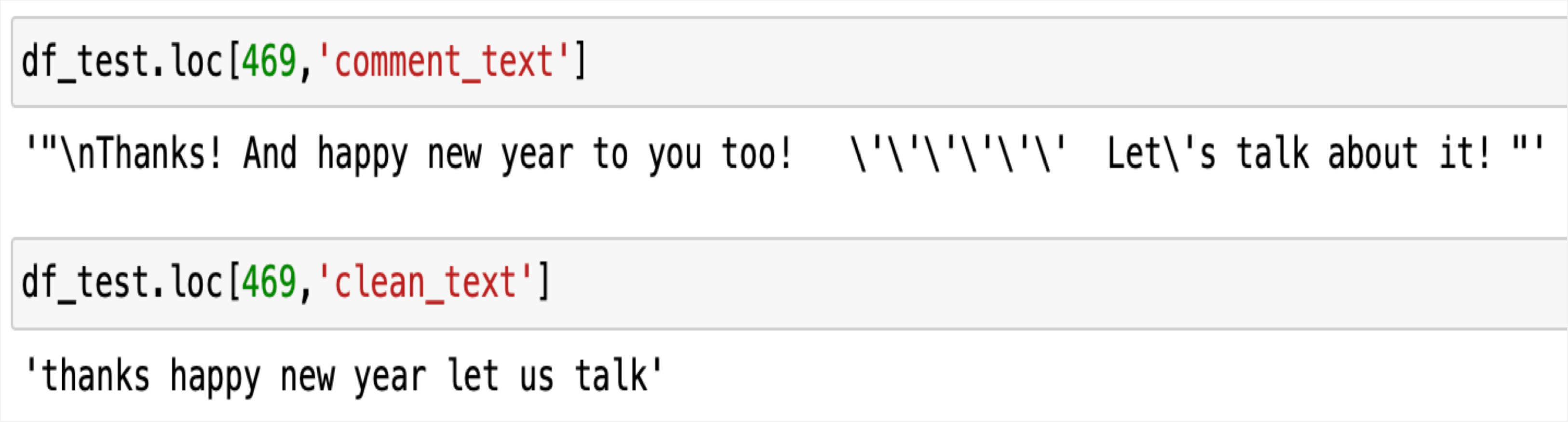
	toxic	severe_toxic	obscene	threat	insult	identity_hate	Count
0	0	0	0	0	0	0	143346
1	1	0	0	0	0	0	5666
2	1	0	1	0	1	0	3800
3	1	0	1	0	0	0	1758
4	1	0	0	0	1	0	1215
5	1	1	1	0	1	0	989
6	1	0	1	0	1	1	618
7	0	0	1	0	0	0	317
8	0	0	0	0	1	0	301
9	1	1	1	0	1	1	265

- Comment counts for the highest 10 combinations
- Most comments with other 5 labels also labeled ‘toxic’

TEXT CLEANING & PREPROCESSING

Text Cleaning and preprocessing pipeline

- Standardize comments to lower-case
- Remove accents from characters
- Remove special characters like emoji’s, punctuations
- Remove contractions
- Lemmatize all words
- Remove stop-words such as ‘a’, ‘you’, ‘the’ to focus on words that are more important to identify the meaning of a comment



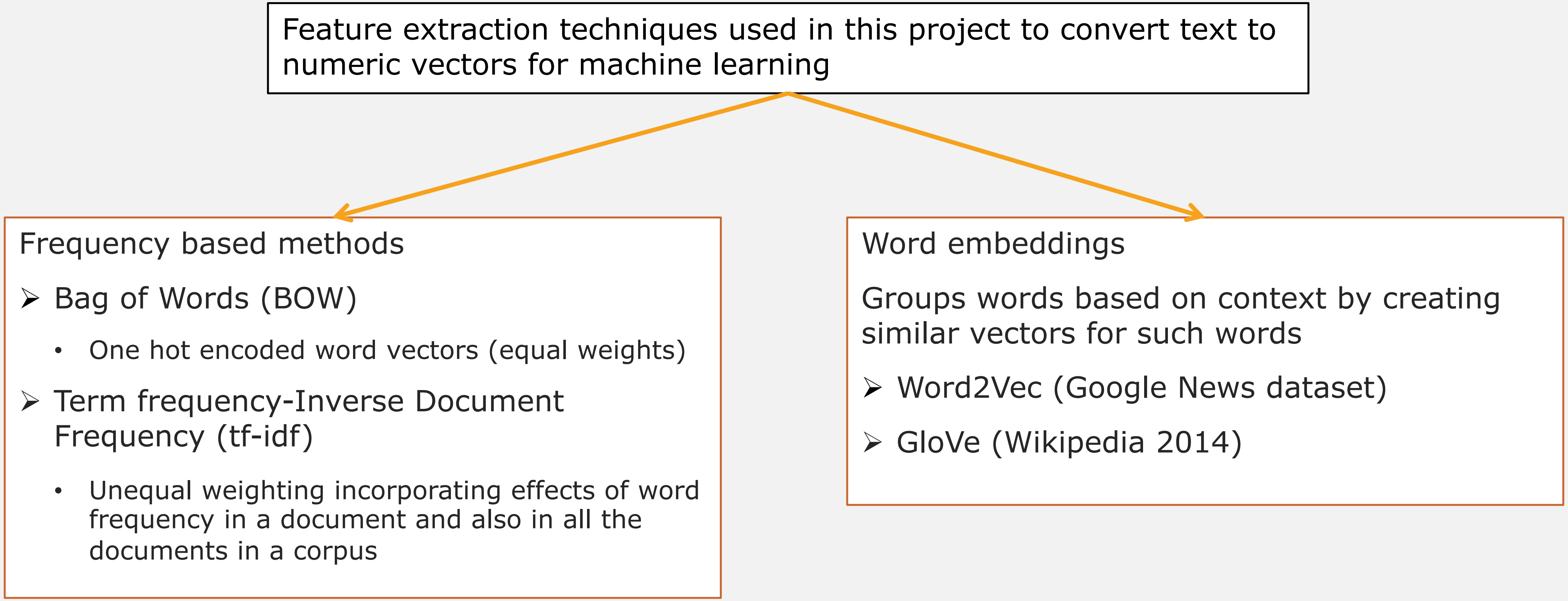
```
df_test.loc[469,'comment_text']
```

```
'"\nThanks! And happy new year to you too!  \'\'\'\'\'\'  Let\'s talk about it! "'
```

```
df_test.loc[469,'clean_text']
```

```
'thanks happy new year let us talk'
```

WORD REPRESENTATION



Feature extraction techniques used in this project to convert text to numeric vectors for machine learning

Frequency based methods

- Bag of Words (BOW)
  - One hot encoded word vectors (equal weights)
- Term frequency-Inverse Document Frequency (tf-idf)
  - Unequal weighting incorporating effects of word frequency in a document and also in all the documents in a corpus

Word embeddings

Groups words based on context by creating similar vectors for such words

- Word2Vec (Google News dataset)
- GloVe (Wikipedia 2014)

MODELING

Configurations

- Bag of Words / Tf-idf with and without resampling to handle label imbalance
- Word2Vec / GloVe averaged vectors with and without resampling to handle label imbalance
- Six individual binary classifiers to predict for each label
- Metrics used: AUC score, F1-score, Hamming loss
- Models:
    - Multinomial Naïve Bayes
    - Logistic Regression
    - Light Gradient Boosting

MODELING
(BAG OF WORDS / TF-IDF)

Without resampling

Model	Word Vector	Mean Train AUC	Mean Test AUC	Mean Train F1	Mean Test F1
Multinomial Naïve Bayes	Bag of Words	0.956	0.935	0.548	0.519
	Tf-idf	0.963	0.951	0.4	0.365
Logistic Regression	Bag of Words	0.987	0.94	0.625	0.487
	Tf-idf	0.988	0.978	0.522	0.509
Light Gradient Boost (LGBM)	Bag of Words	0.99	0.965	0.74	0.535
	Tf-idf	0.993	0.968	0.789	0.533

With resampling (SMOTE)

Model	Word Vector Type	Mean Train AUC	Mean Test AUC	Mean Train F1	Mean Test F1
Multinomial Naïve Bayes	Bag of Words	0.98	0.874	0.827	0.466
	Tf-idf	0.978	0.958	0.879	0.422
Logistic Regression	Bag of Words	0.989	0.881	0.95	0.35
	Tf-idf	0.989	0.963	0.954	0.48
Light Gradient Boost (LGBM)	Bag of Words	0.982	0.919	0.915	0.326
	Tf-idf	0.992	0.969	0.95	0.346

- Similar performances for Bag of Words and Tf-idf
- Resampling led to overfitting
- Logistic Regression with TF-IDF without resampling has optimal performance

MODELING
(WORD2VEC / GLOVE)

Without resampling

Model	Word Vector Type	Mean Train AUC	Mean Test AUC	Mean Train F1	Mean Test F1
Logistic Regression	Word2Vec	0.967	0.963	0.417	0.411
	GloVe	0.963	0.96	0.418	0.419
Light Gradient Boost	Word2Vec	0.989	0.941	0.858	0.458
	GloVe	0.993	0.953	0.868	0.452

With resampling (SMOTE)

Model	Word Vector Type	Mean Train AUC	Mean Test AUC	Mean Train F1	Mean Test F1
Logistic Regression	Word2Vec	0.973	0.964	0.854	0.43
	GloVe	0.969	0.961	0.84	0.414
Light Gradient Boost	Word2Vec	0.995	0.967	0.955	0.54
	GloVe	0.994	0.964	0.95	0.534

- Similar performances between Word2Vec and GloVe
- As the previous slide, resampling led to overfitting
- Logistic regression without resampling had optimal performance
- No performance improvement over frequency methods
  - Potential Cause: 65% (Word2Vec) and 47% (GloVe) of the unique corpus words not in the pre-trained model vocabularies

CONCLUSIONS

- Project aimed at creating multi-label classification models for a comments dataset with six inflammatory language labels
- Bag of Word/TF-IDF & word embeddings Word2Vec/GloVe were used to create word vectors
- Models used: Naïve Bayes, Logistic Regression and Light Gradient Boost
- Resampling with SMOTE caused overfitting with all the models
- Logistic regression with TF-IDF and no resampling had an optimal performance with minimal overfitting
- No model improvement using word embeddings
  - Significant % of unique corpus words not present in the Word2Vec/GloVe vocabularies

FUTURE WORK

- Using sub-word vectors to create comment vectors using models such as FastText to potentially handle spelling mistakes
- Use multi-label modeling techniques such as label powerset or classifier chaining to include correlations between multiple labels
- Apply deep-learning techniques

REFERENCES

1. https://seedscientific.com/how-much-data-is-created-every-day/
2. https://ourworldindata.org/rise-of-social-media
3. https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge
4. https://mccormickml.com/2016/04/12/googles-pretrained-word2vec-model-in-python/
5. https://nlp.stanford.edu/projects/glove/
6. https://stackoverflow.com/questions/49239941/what-is-unk-in-the-pretrained-glove-vector-files-e-g-glove-6b-50d-txt
7. https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
8. https://towardsdatascience.com/a-practitioners-guide-to-natural-language-processing-part-i-processing-understanding-text-9f4abfd13e72
9. https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bff