WebAPI and Classifiers for Tensorflow and Pytorch subreddits

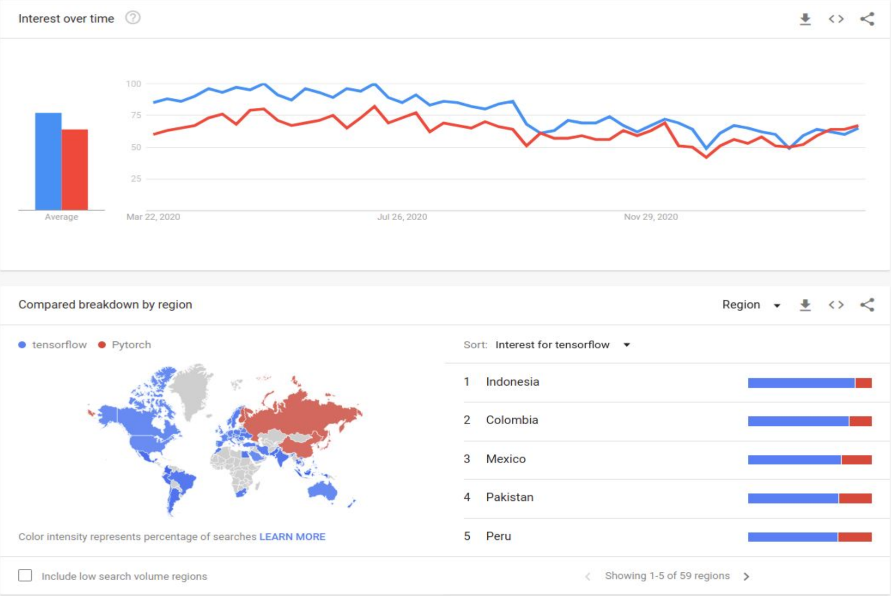
- Zhi Qiang (DSI 20)

Subreddits of interest

Two competing frameworks used for deep learning

- Tensorflow
  - Developed by Google Brain
  - Released in 2015
  - Widely used in production

- Pytorch
  - Developed by Facebook AI Research
  - Released in 2016
  - Widely used in research than in production



1 Indonesia
2 Colombia
3 Mexico
4 Pakistan
5 Peru

Color intensity represents percentage of searches LEARN MORE

□ Include low search volume regions

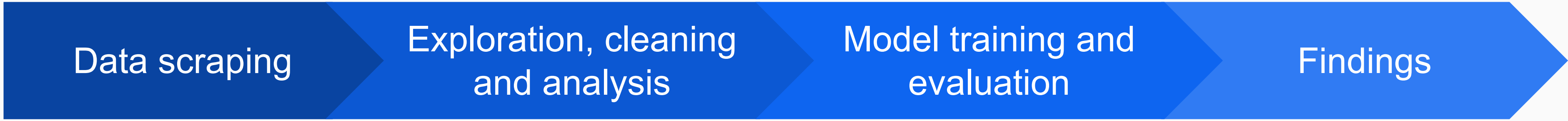
Showing 1-5 of 59 regions

Problem statement

Posting of contents in wrong subreddit is always an issue. As such, the subreddit moderators are required to clean up such posts occasionally to ensure content relevancy for viewers of these subreddits.

By having good classifiers developed to classify erroneous posts, the amount of time by the moderators to clean up would be reduced.

General Workflow



Data scraping

Exploration, cleaning and analysis

Model training and evaluation

Findings

- ~2000 posts scraped from each subreddit

- Duplicates removal
- Identify columns of interest and feature engineering
- Word count and N-gram analysis
- Lemmatization
- Stopwords removal

- Vectorization
- Training multiple classifiers:
  - Naive Bayes
  - Regression
  - Tree ensembles

- Model comparison
- Analysis of misclassified results

Challenges in data

- Large amount of duplicates
  - Additional scraping maybe required
  - Imbalanced class representations after removing duplicates
- Irrelevant metadata information
- Missing text in posts
  - Non-text posts such as images

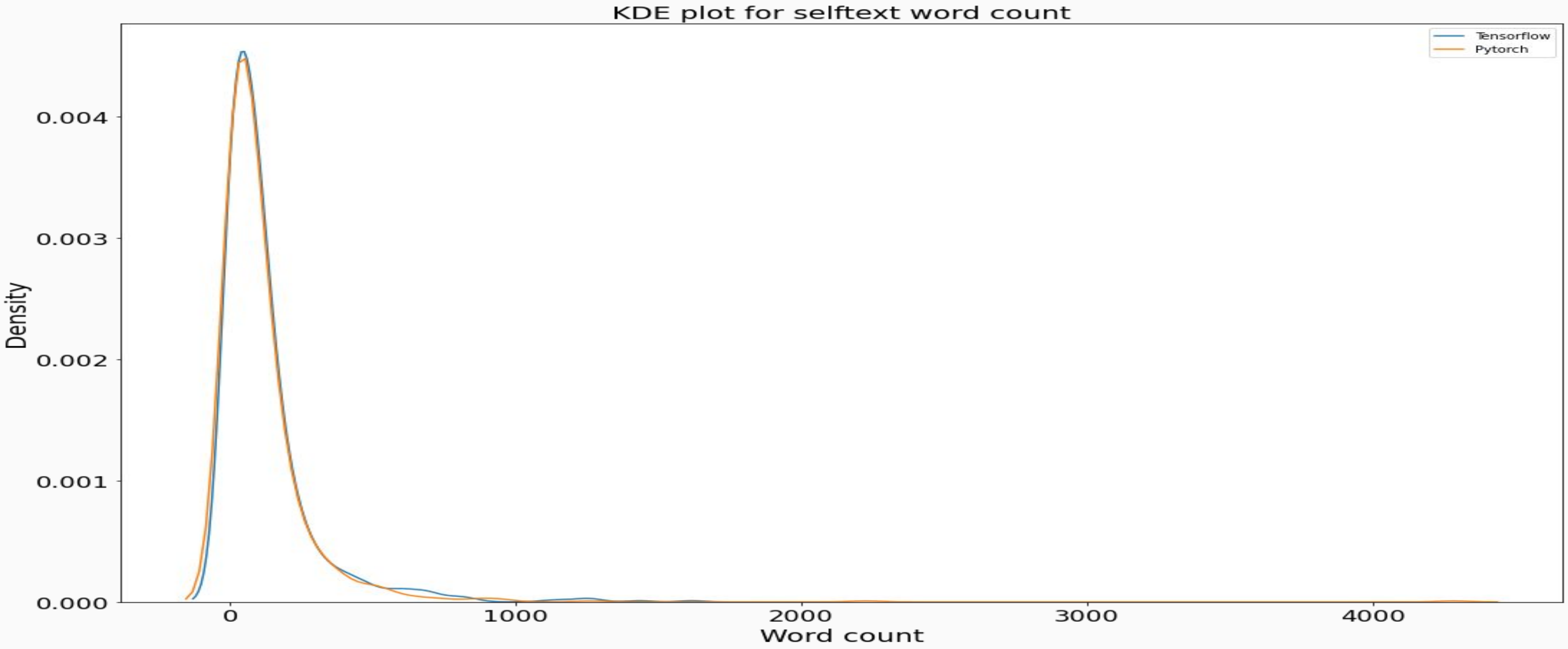
EDA

Data information

	Posts	Posts after removal of duplicates	Empty *selftext* posts
Tensorflow	1992	918 (46%)	207 out of 918 (22.5%)
Pytorch	1988	943 (47.4%)	243 out of 943 (25.8%)

Word length analysis of posts

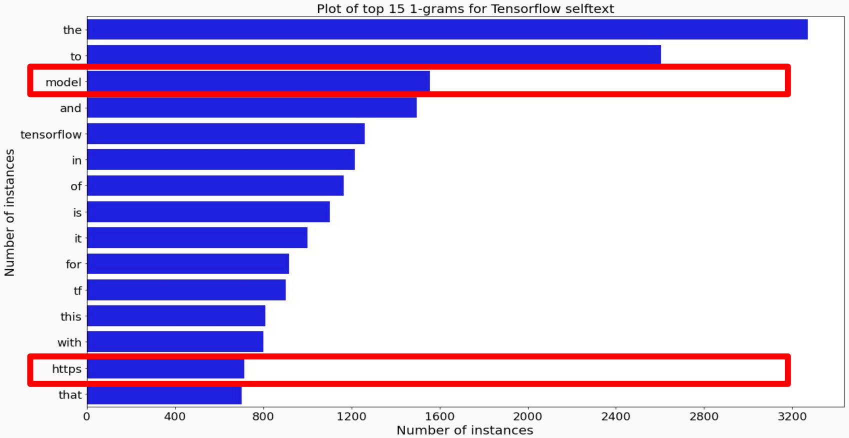
- KDE plot are similar for both subreddits

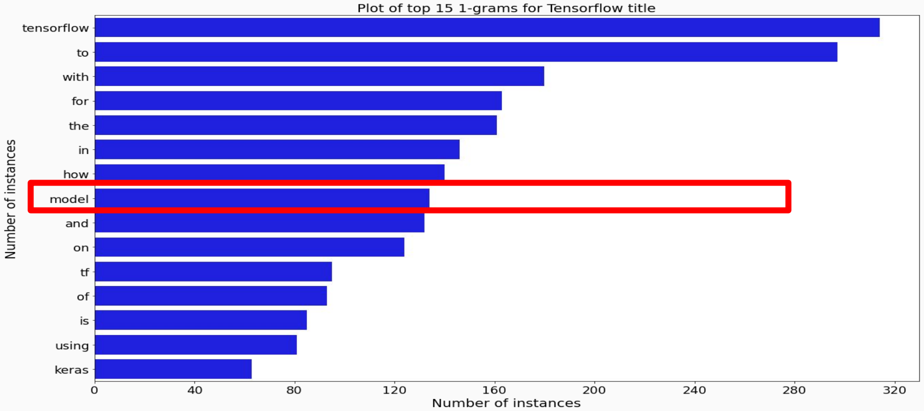


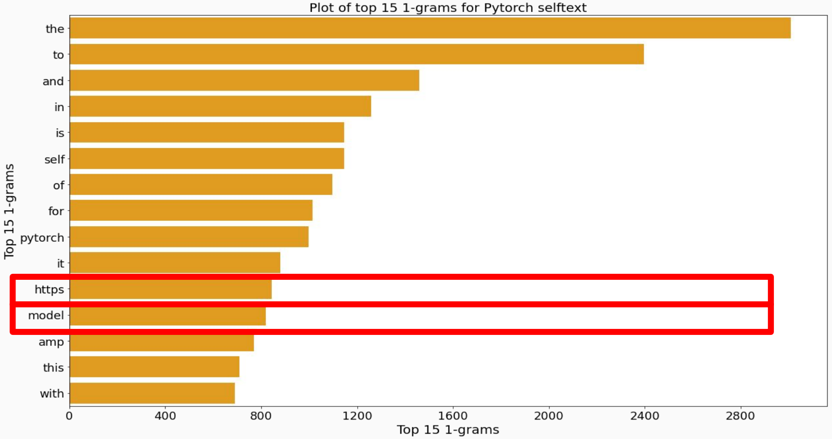
N-gram analysis

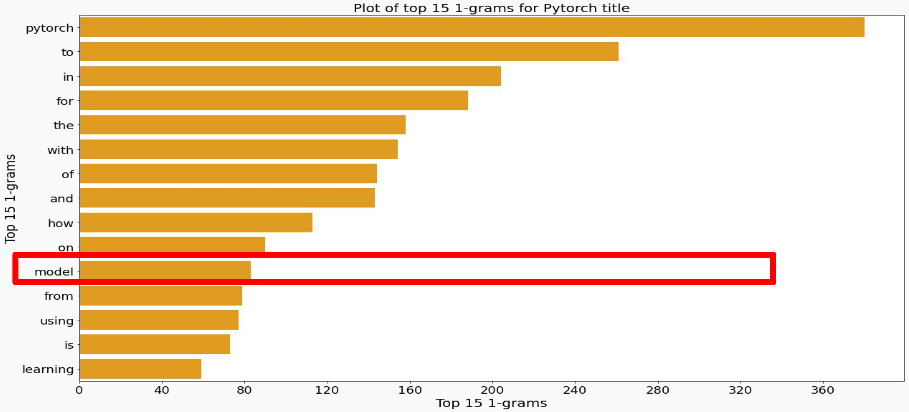
- Identify most frequent words/phrases
  - Possibly include them into stopwords
- N-grams considered:
  - unigram
  - bigram
  - trigram

Unigram analysis of data

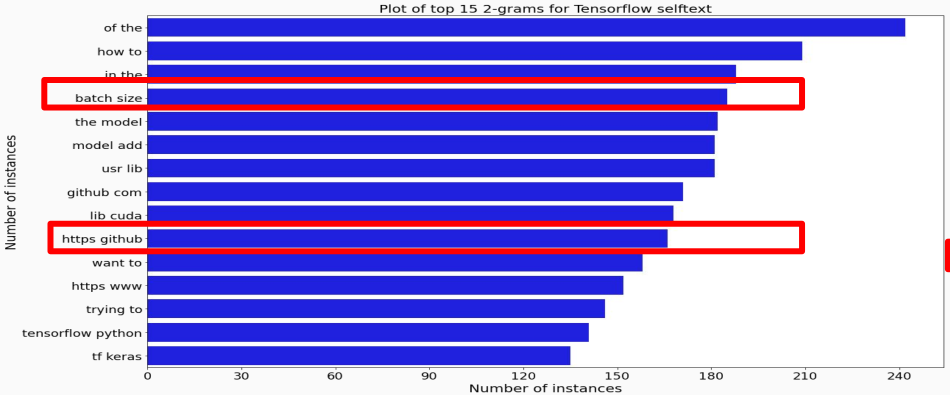

Plot of top 15 1-grams for Tensorflow selftext

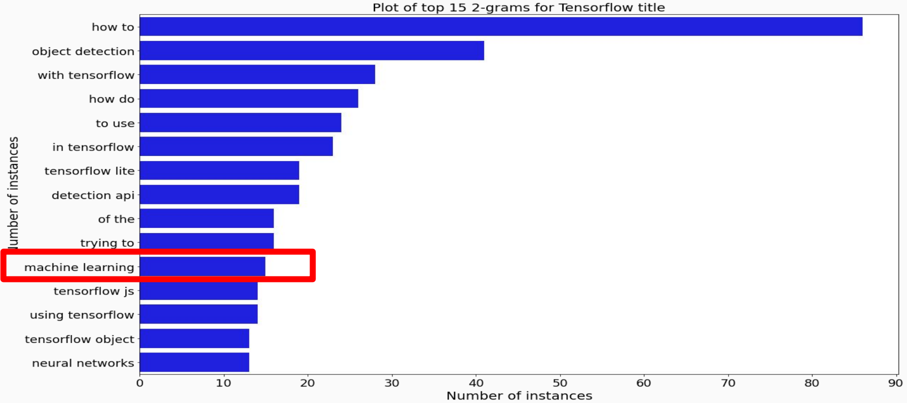

Plot of top 15 1-grams for Tensorflow title

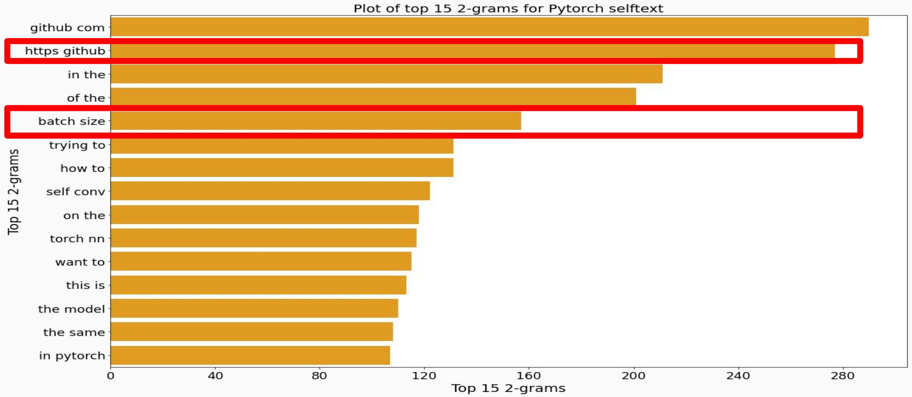

Plot of top 15 1-grams for Pytorch selftext

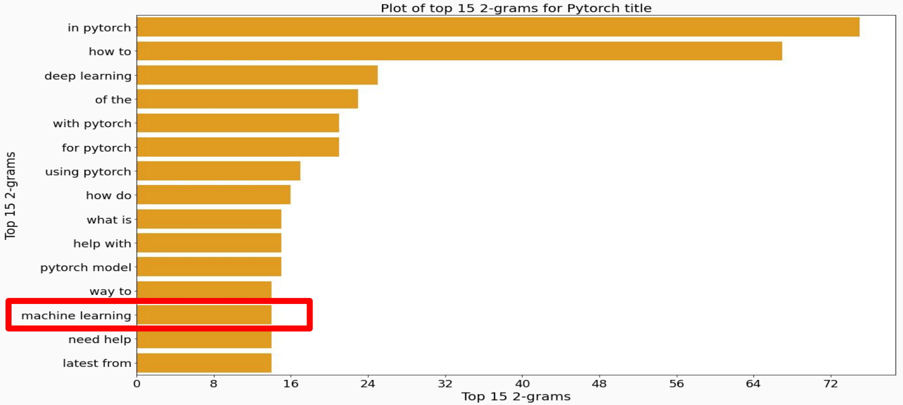

Plot of top 15 1-grams for Pytorch title

Bigram analysis of data

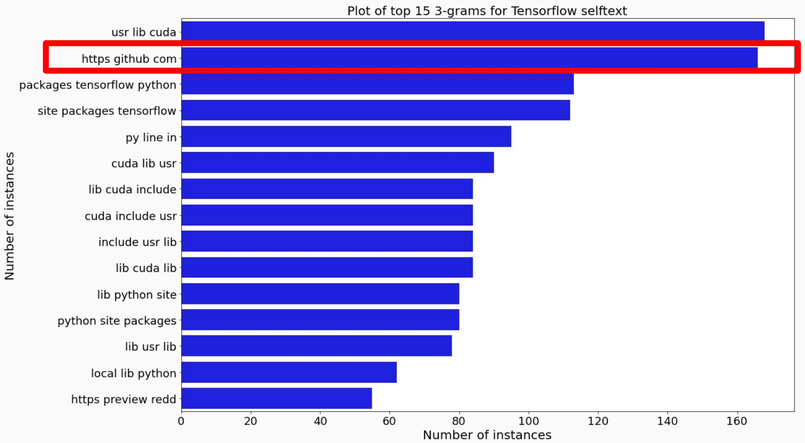


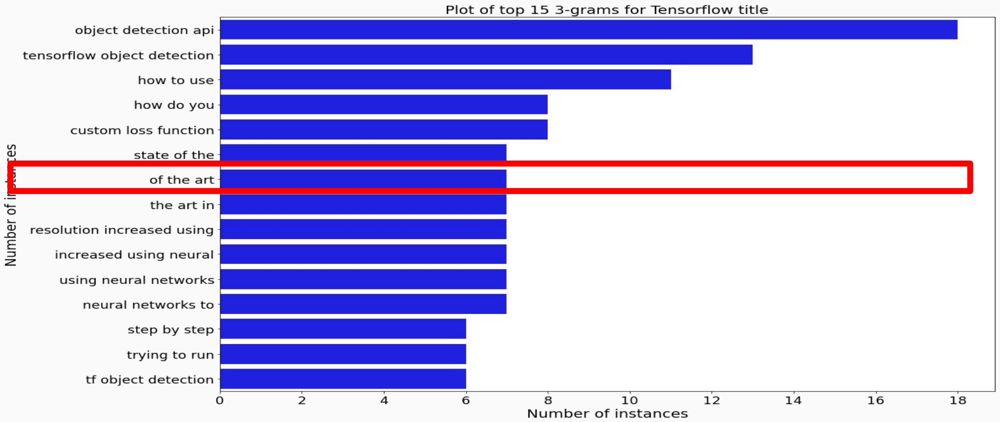


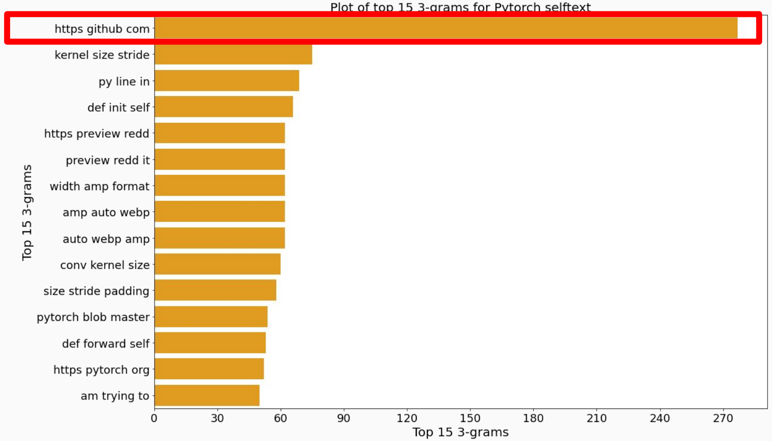


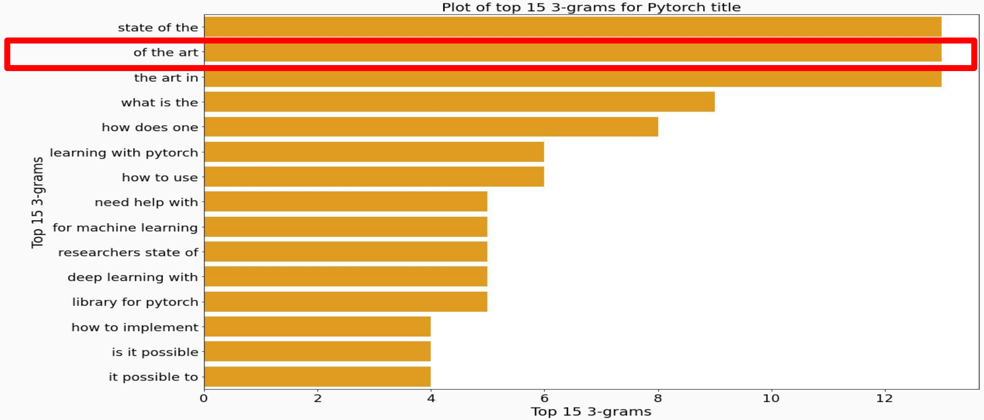


Trigram analysis of data







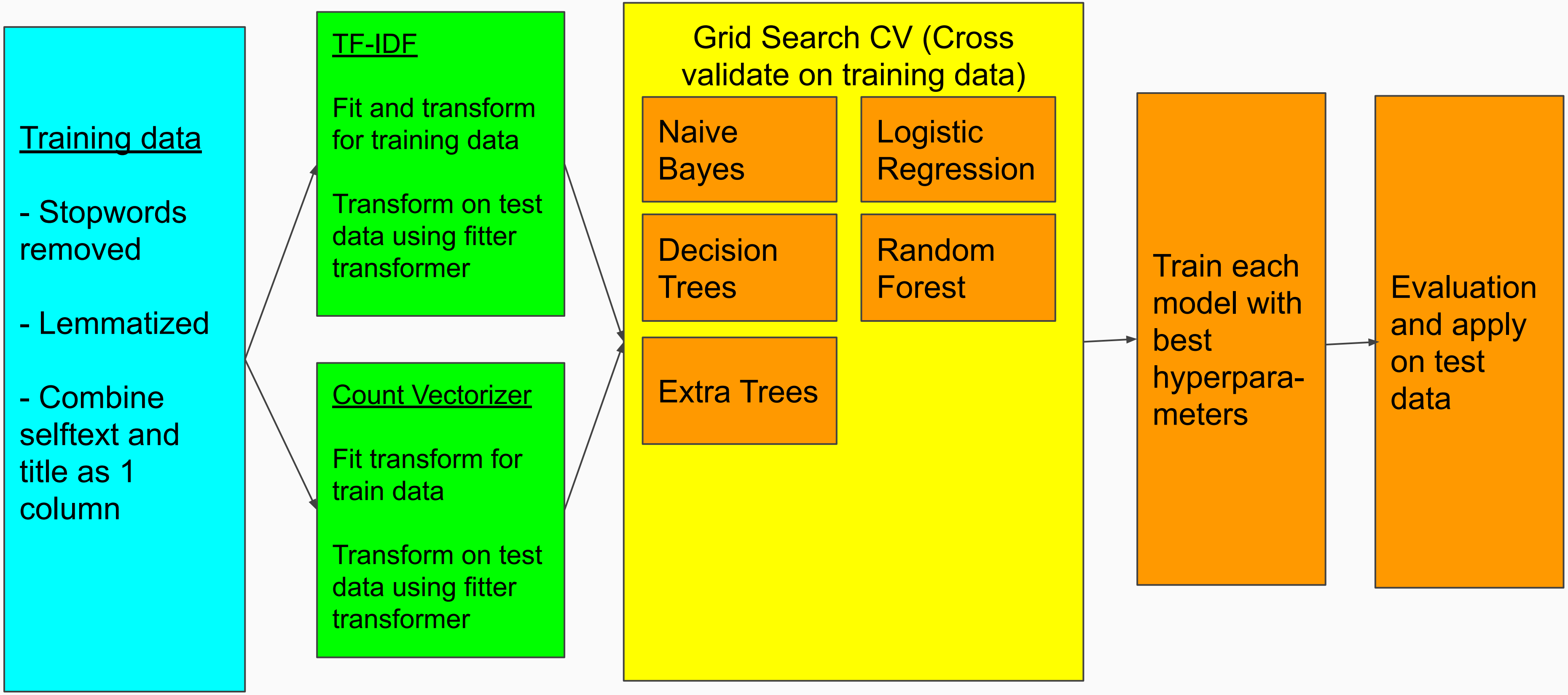


Add words to list of stopwords

- “Https”
- “Github”
- “Com”
- “Machine”
- “Learning”
- “Model”
- “Batch”
- “Size”

Model training, evaluation and comparison

Model training



Training data

- Stopwords removed

- Lemmatized

- Combine selftext and title as 1 column

TF-IDF

Fit and transform for training data

Transform on test data using fitter transformer

Count Vectorizer

Fit transform for train data

Transform on test data using fitter transformer

Grid Search CV (Cross validate on training data)

Naive Bayes

Logistic Regression

Decision Trees

Random Forest

Extra Trees

Train each model with best hyperparameters

Evaluation and apply on test data

Model Performance

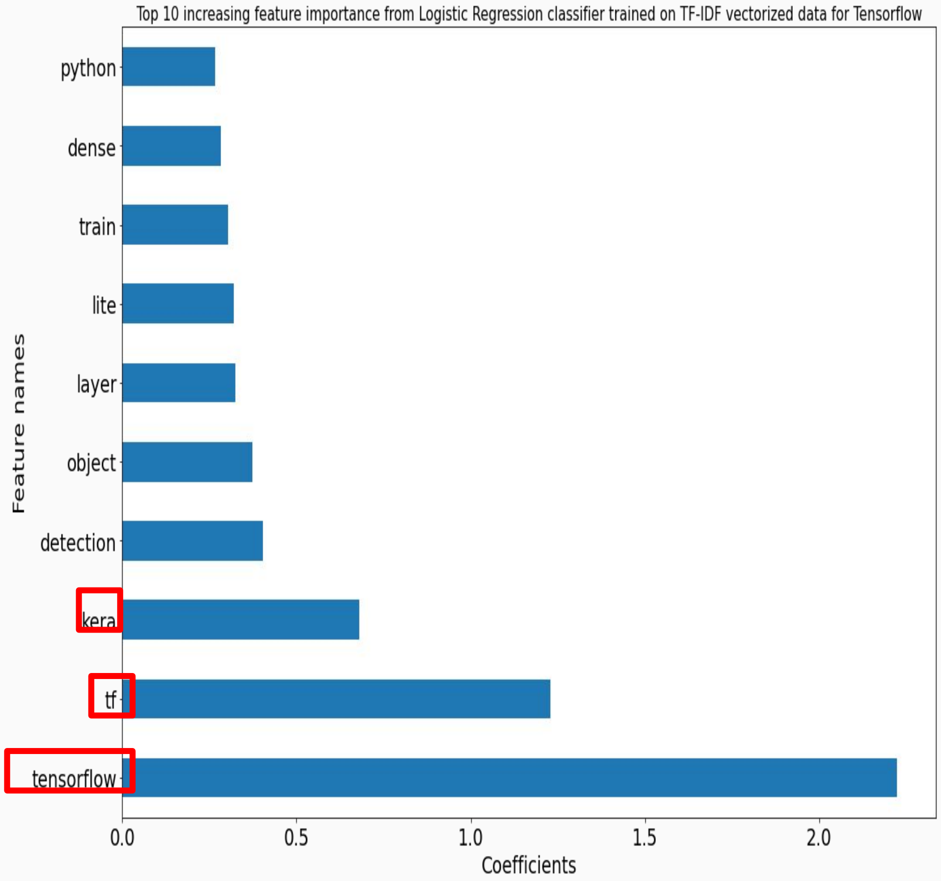
	Count Vectorizer (unigram, 500 features) Baseline 51%		Term Frequency - Inverse Document Frequency Vectorizer (unigram, 500 features) Baseline 51%	
Models	Cross validated training accuracy in %	Test accuracy in %	Cross validated training accuracy in %	Test accuracy in %
Naive Bayes	81.80	75.54	81.80	75.54
Log Regression	85.74	83.91	85.60	85.41
Decision Trees	84.74	81.76	84.81	53.86
Random Forests	87.03	82.19	86.10	82.62
Extra Trees	84.66	85.62	85.38	84.33

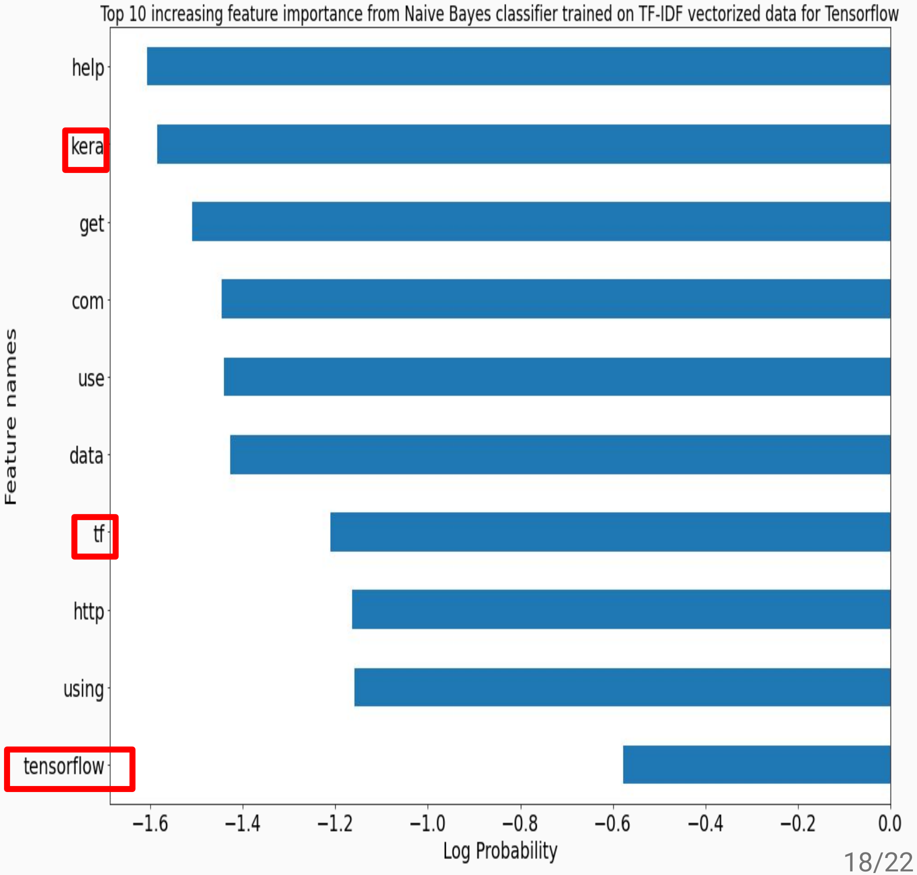
Model to compare with Naive Bayes

- Logistic Regression model is a better model compared to Naive Bayes
  - Close accuracy scores suggests nice fitting of model
- A look at the summary of confusion matrix:

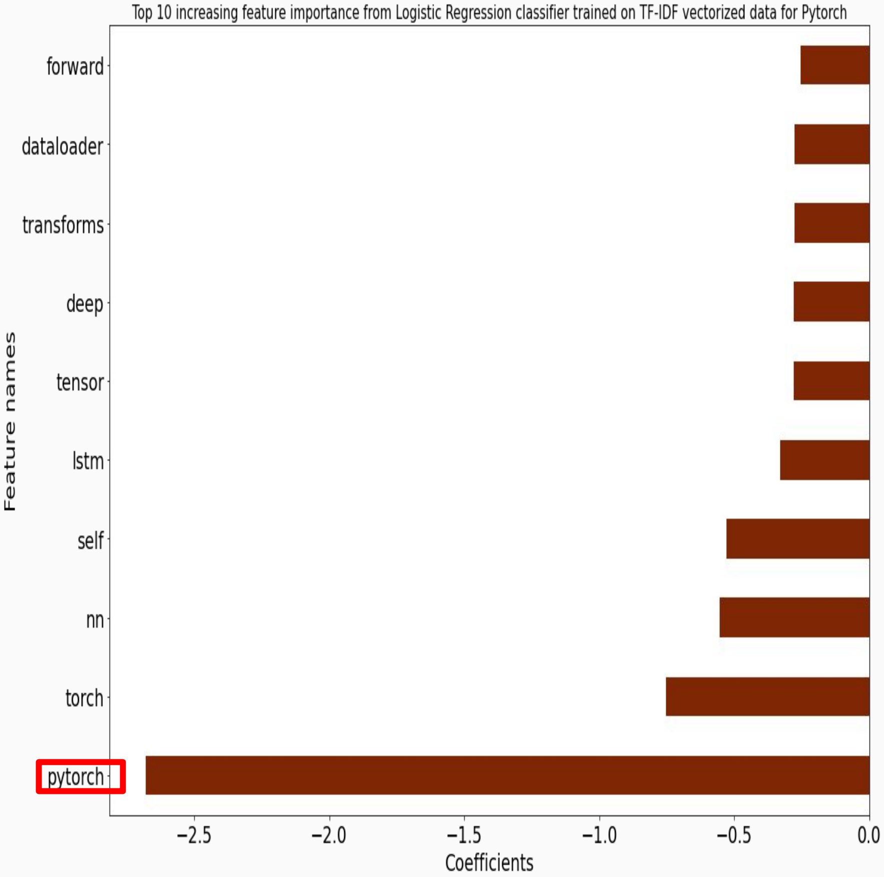
Models	Count Vectorizer (unigram, 500 features) On 466 test data (25%)				Term Frequency - Inverse Document Frequency (unigram, 500 features) On 466 test data (25%)			
Confusion matrix metrics	True Positives (tensorflow)	True Negatives (pytorch)	False Positives	False Negatives	True Positives (tensorflow)	True Negatives (pytorch)	False Positives	False Negatives
Naive Bayes	143	209	27	87	143	209	27	87
Log Regression	167	224	12	63	183	215	21	47

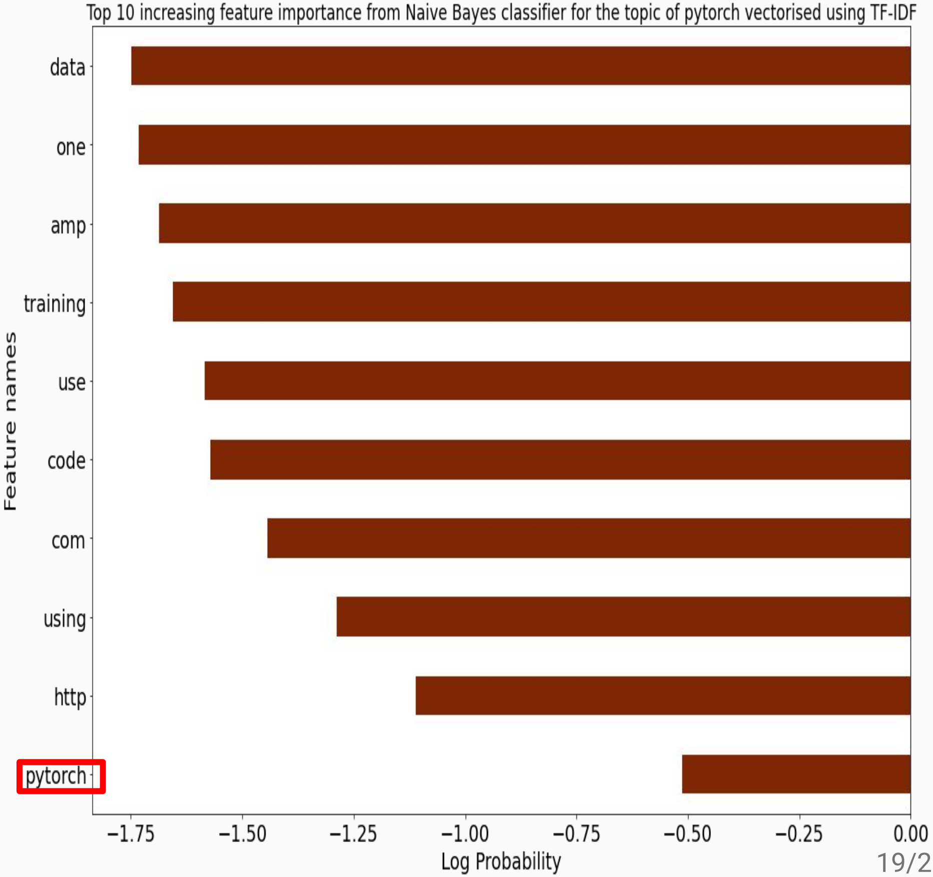
Feature importance for Tensorflow: Logistic Regression vs Naive Bayes





Feature importance for Pytorch: Logistic Regression vs Naive Bayes





Possible causes for misclassification

- Absence of key terms
  - Tensorflow, pytorch
- Short forms - reduced effects
  - Tensorflow as tf
- Non text posts

Selftext	Title	Possible cause of misclassification
have people been using deep learning to do regression i noticed that fitting polynomials using least squares leads to much better accuracy is there any rule of thumb to get arbitrary accuracy with deep regression	deep regression	Absence of key terms
-	what can i do with such a dataset can i use it to predict sales any ideas	Lack of text posts as indicated with -
-	tf based animation	Use of short form “tf” and lack of text posts as indicated with -

Possible improvements

- Replace short form expressions
- Comments to be considered
- Add in more stopwords
  - E.g Neural networks
- Consider different vectorizations and different n-grams
- Consider other models

End