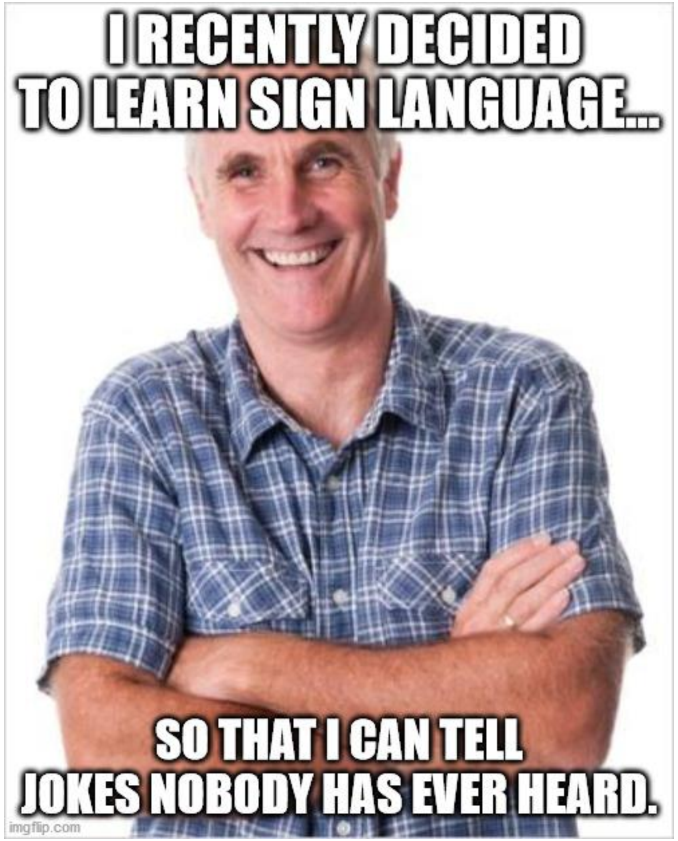
Classifying Dadjokes vs Antijokes

Yuanfeng

Table Of Content

What is a dadjoke?

- A wholesome short joke typically told by fathers with a punchline that is often an obvious or predictable pun or play on words.
- Dadjokes are usually inoffensive, told with sincere humorous intent and more accepted by the public.



What is an antijoke?

- A joke that starts like a standard joke, but then turns out not to be a joke at all.
    - The surprise element thus becoming the joke.
- Antijokes may be more offensive, and the format is less accepted by the public as Dadjokes.

Aim of Project

This project aims to assist any services that need to curate and pick dadjokes from a large number of jokes containing both dadjokes and antijokes.

The **aim of this project** will be to:

- Create a model that classifies if a joke is a dadjoke
- Find out what the most deterministic words for dadjokes and antijokes

Data Informationn

This projects makes use of two datasets scrapped from reddit.

- r/antijokes: 942 records
- r/dadjokes: 1528 records

	subreddit	original_title	original_post	url
0	AntiJokes	You know what they say about black guys in bed	they are in a bed	https://www.reddit.com/r/AntiJokes/comments/k0...
1	AntiJokes	What's an octopus' favorite month?	Despite being an extraordinarily brilliant spe...	https://www.reddit.com/r/AntiJokes/comments/k0...
2	AntiJokes	What do you call a melted snowman?	Water	https://www.reddit.com/r/AntiJokes/comments/k0...
3	AntiJokes	What did the ice cream say to the old man	Jesus fuck I just want an upvote I don't even ...	https://www.reddit.com/r/AntiJokes/comments/jz...
4	AntiJokes	A bartender walks into a bar	He gets working	https://www.reddit.com/r/AntiJokes/comments/k0...

Data Cleaning and Processing

Dropping Null Values

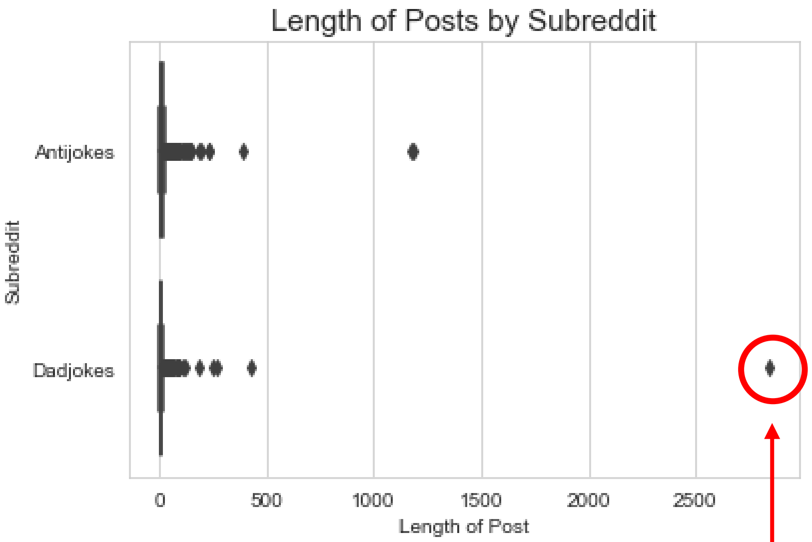
```
# find records with no posts
anti_df.isnull().sum()

# drop the record with no post
anti_df.drop(index=anti_df[anti_df['original_post']
                           .isnull() == True].index, inplace=True)

# reset the index for ease of reference by index
anti_df.reset_index(drop=True, inplace=True)

# check to ensure that there is no more null values
anti_df.isnull().sum()
```

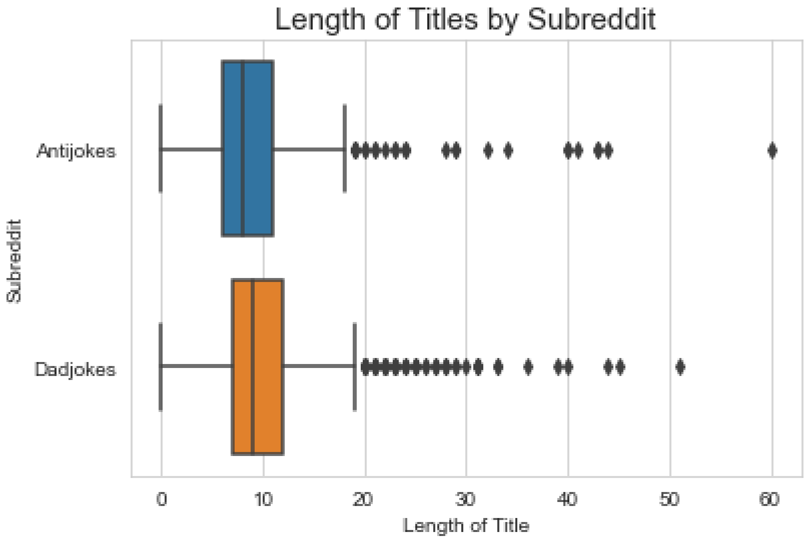
Basic Text Processing

```
# define text processing function
def basic_text_processing(text):

    # remove non-letters
    letters_only = re.sub("[^a-zA-Z]", " ", text)

    # convert each word to lower case
    words = letters_only.lower().split()

    # instantiate lemmatizer
    lemmatizer = WordNetLemmatizer()

    # lemmatize tokens and remove the word 'dad' and 'anti'
    lemmatized_words = [lemmatizer.lemmatize(word) for word in words]

    # remove the word 'dad' and 'anti'
    cleaned_words = [word for word in lemmatized_words
                     if word not in ['dad', 'anti']]

    return (" ".join(cleaned_words))
```

Stopwords were not removed as I will use it as a hyperparameter subsequently

Removing Outlier





Huge outlier - Giant List of Puns

Removing Outlier

Titles length: Majority less than 25.

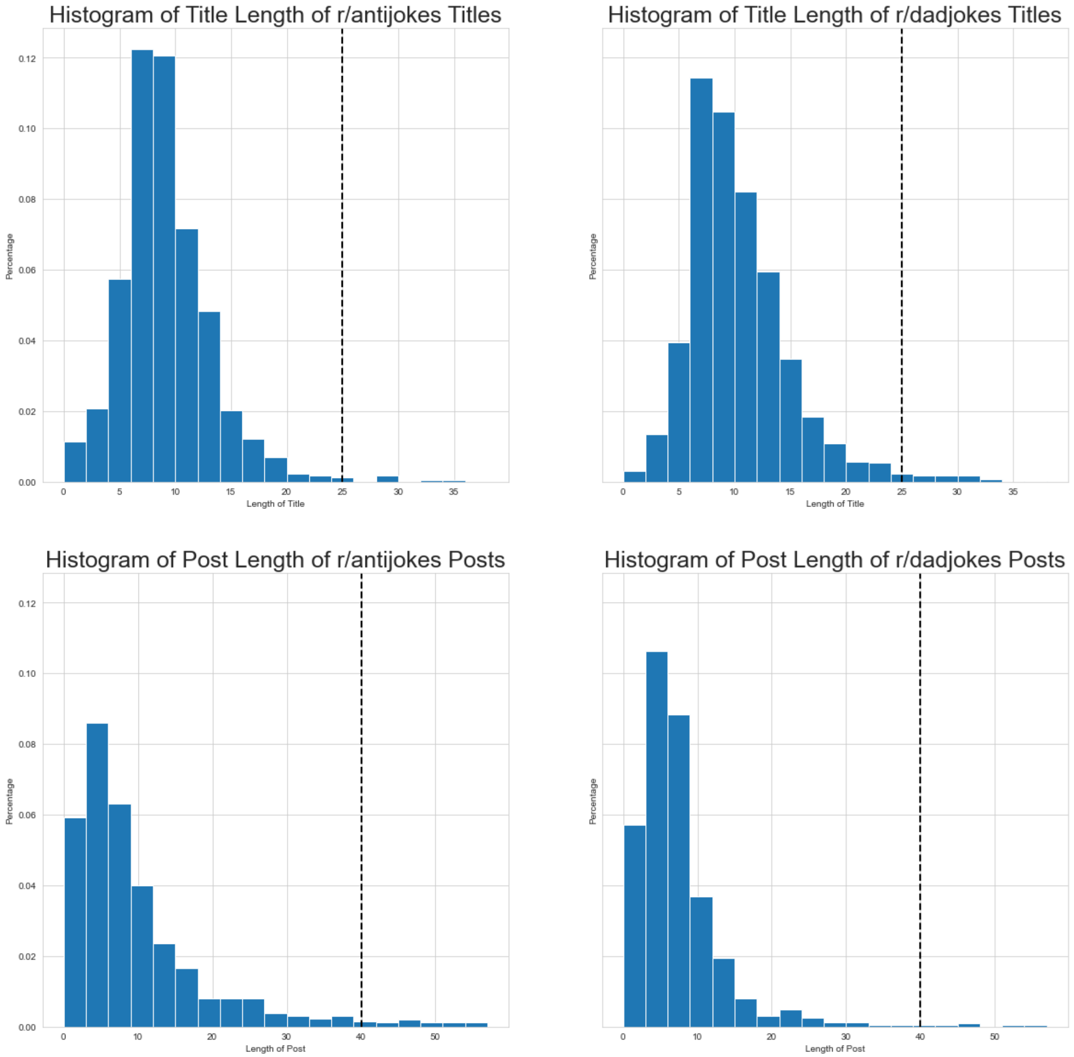
Posts length: Majority less than 40.

Chose:

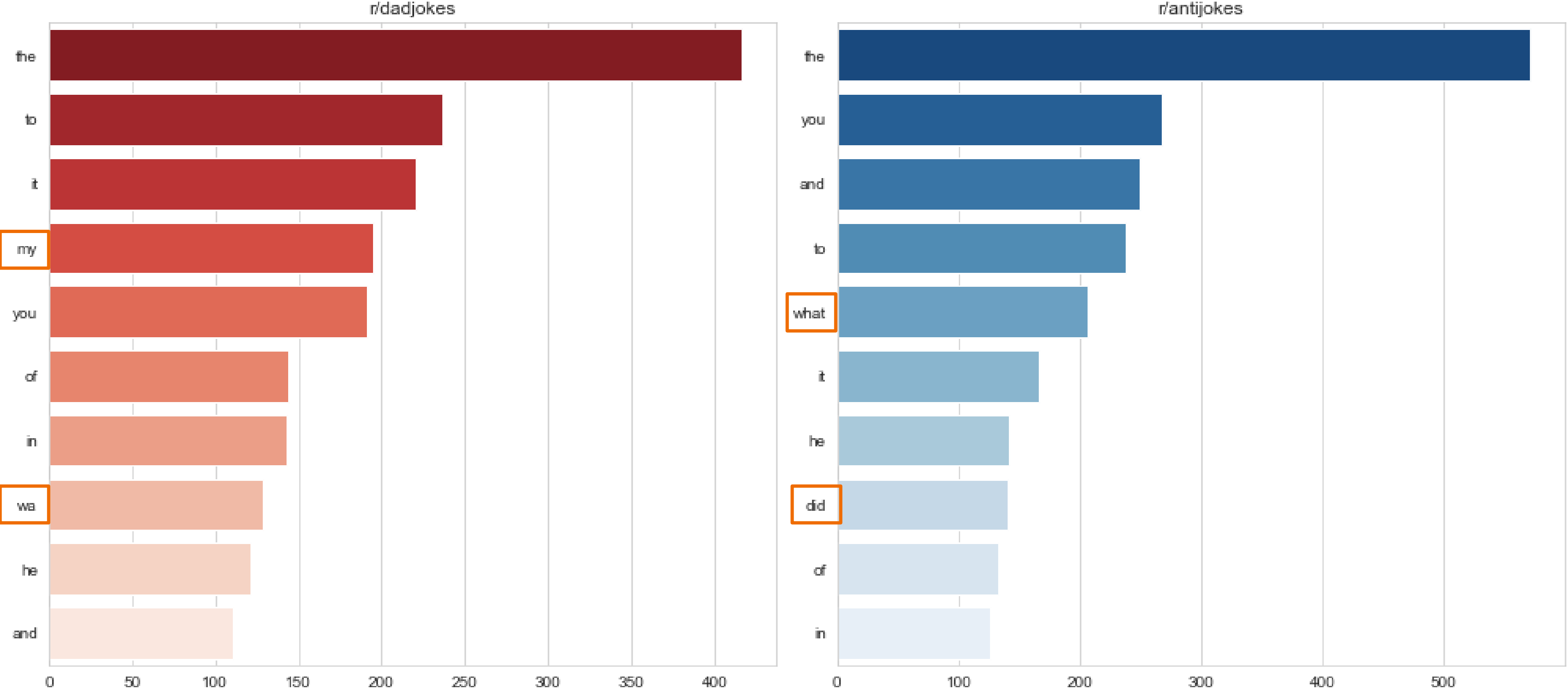
- Title between 3 to 25 words
- Post between 3 to 40 words

Datasets remaining:

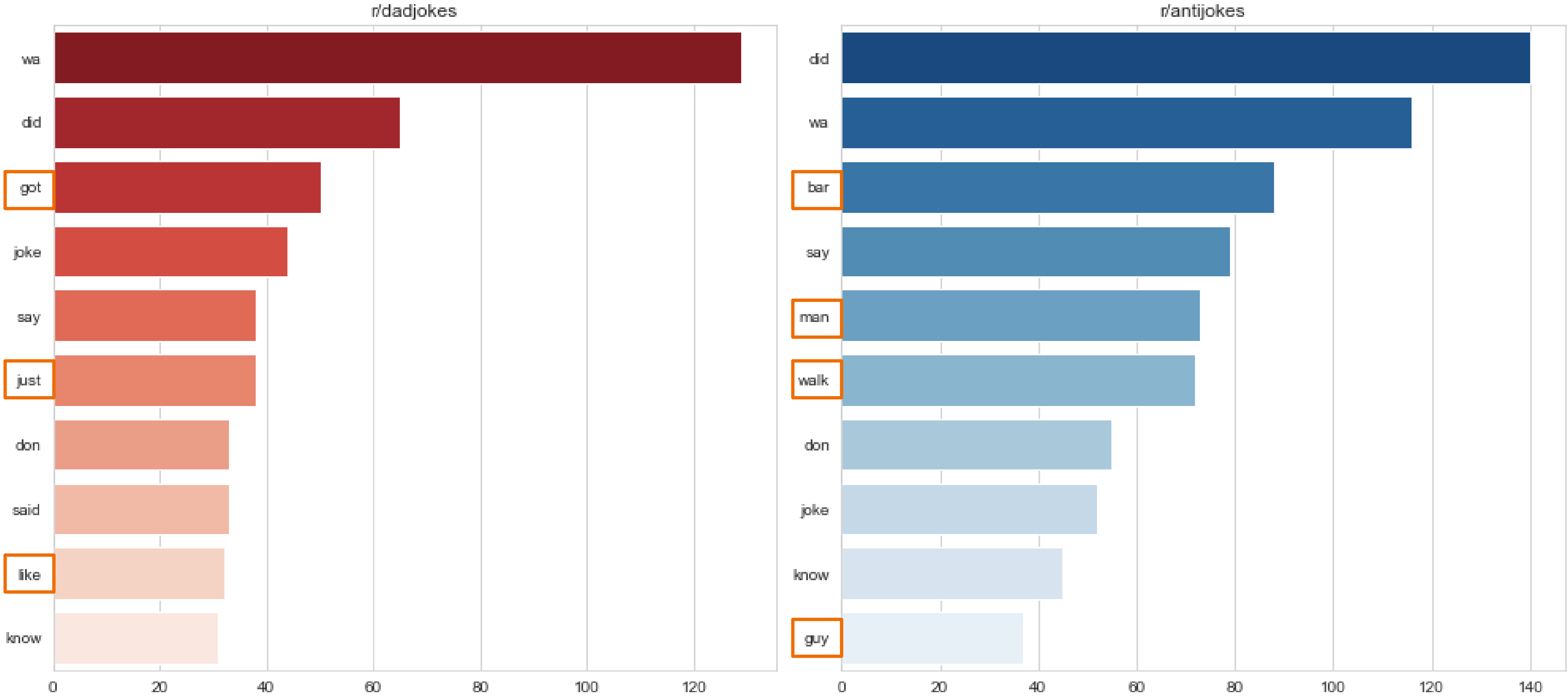
- Antijokes: 649 records
- Dadjokes: 1192 records (sampled 650 records)



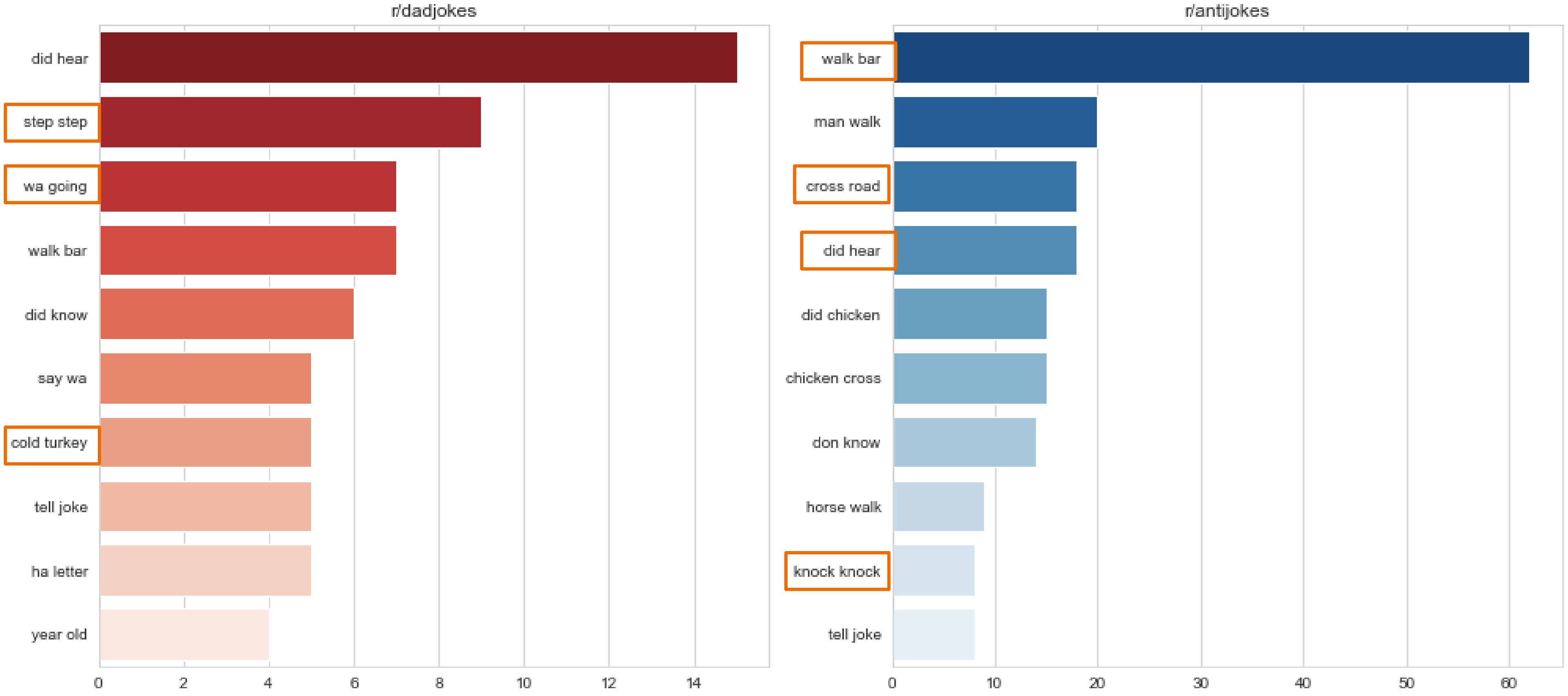
Top 10 Words (Stopwords NOT Removed)



Top 10 Words (Stopwords Removed)



Top 10 Bi-gram Words (Stopwords Removed)




HERE'S A STEP-BY-STEP
GUIDE ON HOW TO FALL DOWN STAIRS!
STEP 28, STEP 27, STEP 24, STEP 21,
STEP 16, STEP 12, STEP 7, STEP 3, STEP 1
imgflip.com

Observations from EDA

- Most of the words with high frequency are common words
- Antijokes have more "man walk into a bar" jokes than dadjokes.
- The difference between dadjokes and antijokes is in the context of the jokes
  - Models is unable to comprehend

Classification Metric

$$F_1 = \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

Classification Metrics Used: F1-score

Interested to find:

- The highest amount of True Positives (accurately predicted dadjokes)
- The least amount of False Positives (inaccurately predicted dadjokes)
- The least amount of False Negatives (inaccurately predicted antijokes)

F1-score is the harmonic mean of Precision and Recall ⇒ best metric to optimize

Models to be used: Logistic Regression, Multinomial NB and Random Forest

Model Results with No Hyperparameter Tuning

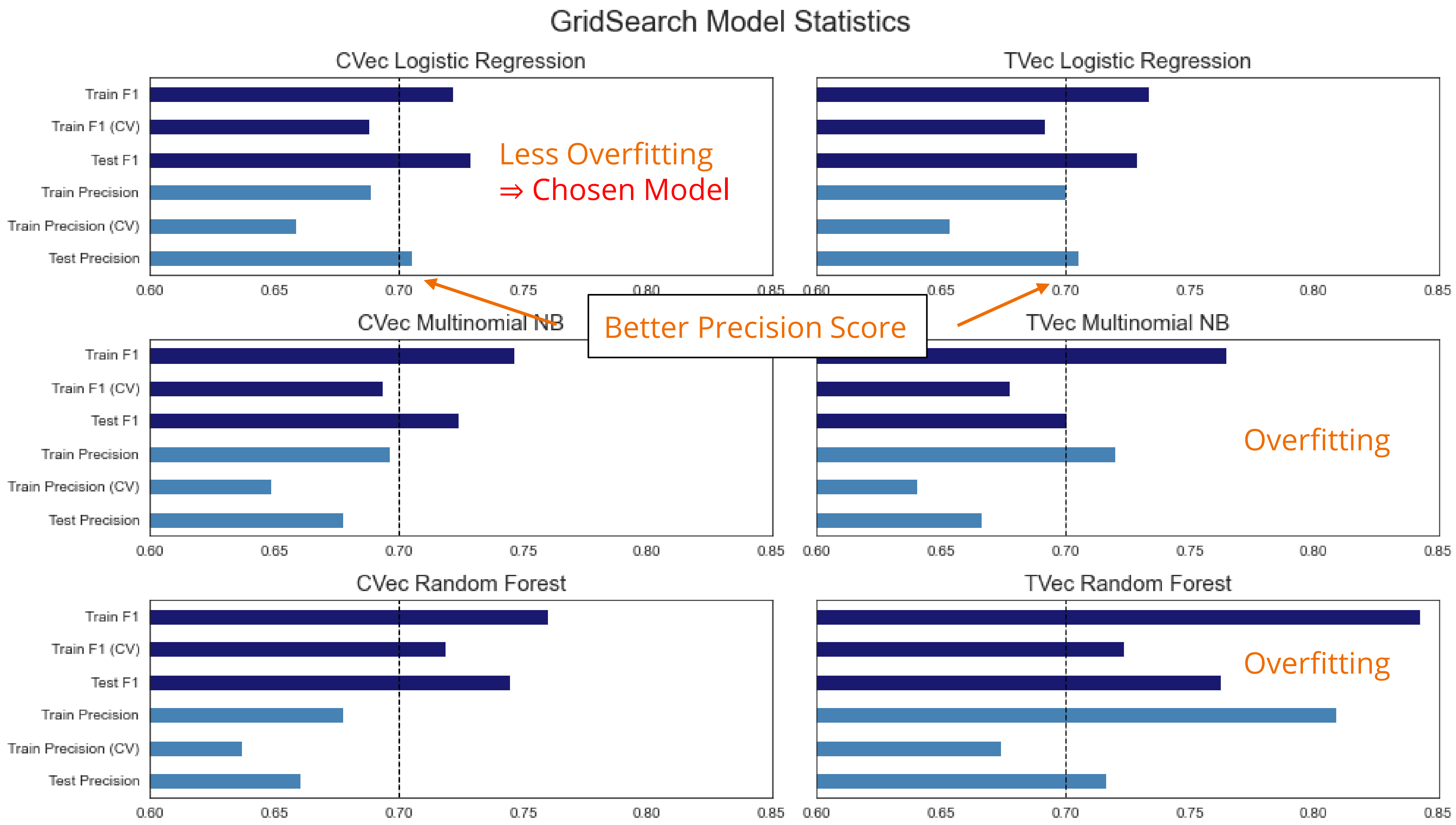
- Ran models with just base vectorizer and model with no hyperparameter gridsearch.
- Extremely overfitted - ~0.3 difference in scores for training dataset and cross validated score on the same training dataset

	Train F1	Train F1 (CV)	Test F1	Train Precision	Train Precision (CV)	Test Precision
CVec Logistic Regression Pipeline	0.9885	0.7140	0.6950	0.9866	0.7194	0.6977
CVec Multinomial NB Pipeline	0.9478	0.6636	0.6396	0.9533	0.7286	0.7717
CVec Random Forest Pipeline	1.0000	0.7163	0.7212	1.0000	0.6946	0.6978
TVec Logistic Regression Pipeline	0.9261	0.6891	0.6911	0.9123	0.7046	0.7328
TVec Multinomial NB Pipeline	0.9543	0.6551	0.6244	0.9646	0.7460	0.7582
TVec Random Forest Pipeline	1.0000	0.6833	0.6923	1.0000	0.6822	0.6923

Model Results with Hyperparameter Tuning

- Ran models with hyperparameter gridsearch.
- Overfitting reduced - Difference reduce from 0.3 to 0.05 for most models.

	Train F1	Train F1 (CV)	Test F1	Train Precision	Train Precision (CV)	Test Precision
CVec Logistic Regression GridSearch	0.7216	0.6882	0.7286	0.6888	0.6588	0.7050
CVec Multinomial NB GridSearch	0.7464	0.6933	0.7240	0.6967	0.6486	0.6779
CVec Random Forest GridSearch	0.7601	0.7187	0.7450	0.6777	0.6369	0.6607
TVec Logistic Regression GridSearch	0.7338	0.6919	0.7286	0.6998	0.6532	0.7050
TVec Multinomial NB GridSearch	0.7647	0.6777	0.7007	0.7199	0.6406	0.6667
TVec Random Forest GridSearch	0.8424	0.7238	0.7626	0.8088	0.6743	0.7162


GridSearch Model Statistics
CVec Logistic Regression
TVec Logistic Regression
CVec Multinomial NB
TVec Multinomial NB
CVec Random Forest
TVec Random Forest
Train F1
Train F1 (CV)
Test F1
Train Precision
Train Precision (CV)
Test Precision
0.60
0.65
0.70
0.75
0.80
0.85
Less Overfitting
⇒ Chosen Model
Better Precision Score
Overfitting
Overfitting

Production Model Statistics

The model was re-trained on the entire dataset.

The final classification model performs better than baseline prediction.

- Precision score = 0.658
  ⇒ Translates to up to 24% reduction in time spent by employee picking dadjokes

F1-Score	0.686
Precision	0.658
Recall	0.717

Cross Validated Result on the Training Dataset

Effect of Top 20 Words

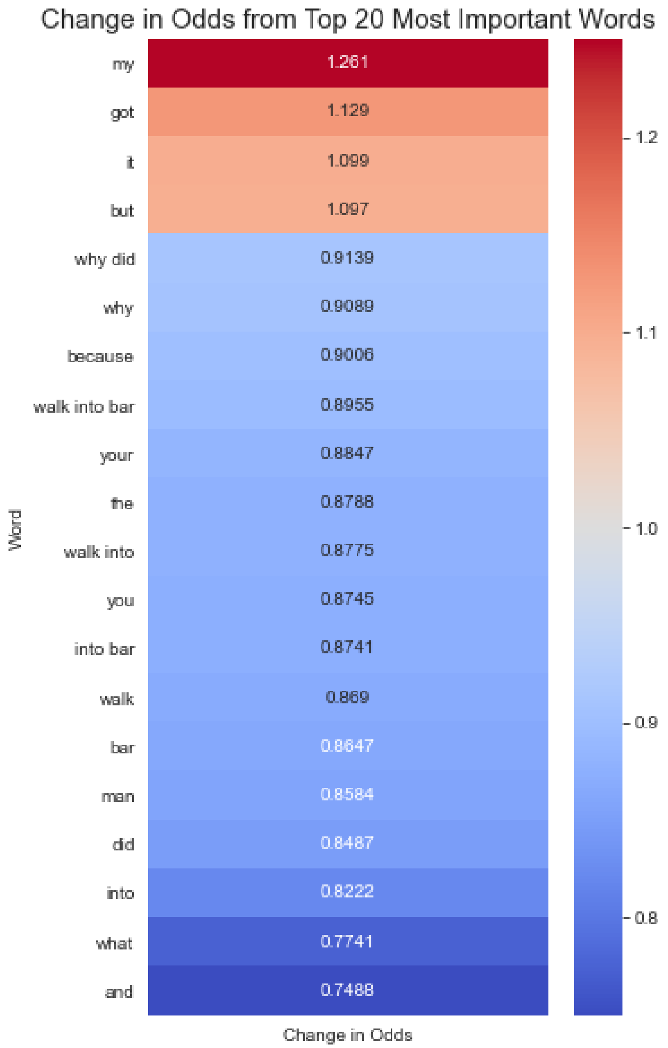
Among the top 20 words:

- Only 4 of the words are identifiers of dadjokes
- Even these 4 words results in very small increase in odds

Identifying words for **dadjokes** are relatively common words such as: "my", "got", "it" & "but".

Identifyinng words for **antijokes** also contains common words such as: "and", "what", "man", "you" & "the".

As expected, jokes with "walked in a bar" were strong predictors of antijokes.



Conclusion

The classification model does not have very high precision as:

- Both type of jokes use similar common English words (not much specialized words)
- Whether a joke is a dadjoke or antijoke is very context based

To improve the model:

1. More sophisticated techniques that tries to explore the context of the text could be used
   a. I.e. POS tagging
2. With more records/data, it will help to improve the model to generalize better.
   a. As there are only about 650 records per dataset, it is easy for the frequency of words to be affected by 1 or 2 entries. (i.e. high frequency of “step step” in bi-gram)

Thank you!