Name: Faris Chaudhry
Batch: LISUM25

Hate Speech Detection

Week 9

Team Member Details

Name: Faris Chaudhry

Email: faris.chaudhry@outlook.com

Country: United Kingdom

University: Imperial College London

Specialization: NLP

Problem Description

“The term hate speech is understood as any type of verbal, written or behavioural communication that attacks or uses derogatory or discriminatory language against a person or group based on what they are, in other words, based on their religion, ethnicity, nationality, race, colour, ancestry, sex or another identity factor.

Hate Speech Detection is generally a task of sentiment classification. So, for training, a model that can classify hate speech from a certain piece of text can be achieved by training it on a data that is generally used to classify sentiments. We will use the Twitter tweets to identify tweets containing Hate speech.”

Name: Faris Chaudhry
Batch: LISUM25

Data Validation

```
train_df.info()
✓ 0.0s

<class 'pandas.core.frame.DataFrame'>
Index: 31962 entries, 1 to 31962
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   label   31962 non-null  int64
 1   tweet   31962 non-null  object
dtypes: int64(1), object(1)
memory usage: 749.1+ KB
```

```
test_df.info()
✓ 0.0s

<class 'pandas.core.frame.DataFrame'>
Index: 17197 entries, 31963 to 49159
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   tweet   17197 non-null  object
dtypes: object(1)
memory usage: 268.7+ KB
```

Validation done:

- Standardise column spaces (lowercase, underscores instead of spaces).
- Remove duplicate tweets, keeping the last one (extra tweets of the same kind don't benefit model).
- Remove rows with null tweets.
- Remove unlabelled data in training set.
- Rename duplicate or null index by reindexing the whole set (this wasn't required in practice).

id	label	tweet
1	0	@user when a father is dysfunctional and is s...
2	0	@user @user thanks for #lyft credit i can't us...
3	0	bihday your majesty
5	0	factsguide: society now #motivation
6	0	[2/2] huge fan fare and big talking before the...
...	...	...
31958	0	ate @user isz that youuu?ðŸ□□□ðŸ□□□ðŸ□□□ðŸ□□□ðŸ□□□ð...
31959	0	to see nina turner on the airwaves trying to...
31960	0	listening to sad songs on a monday morning otw...
31961	1	@user #sikh #temple vandalised in in #calgary,...
31962	0	thank you @user for you follow

29530 rows × 2 columns

Name: Faris Chaudhry
Batch: LISUM25

Feature Extraction

Features extracted:

- Word count
- Average word length (sometimes people forget spaces when typing angrily)
- Hashtags
- Exclamation marks
- Question marks (people ask rhetorical questions when angry at someone e.g., 'Are you stupid?")
- All uppercase words (indicates rage)

Correlation was checked on these simple features since if there were multiple features with strong correlation it would only add extra dimensions to the model.

	label	word_count	avg_word	hashtags	exclamation_marks	question_marks	upper
label	1.000000	0.006989	-0.017643	-0.036279	-0.051845	0.045673	0.007141
word_count	0.006989	1.000000	-0.354335	0.016857	0.003394	0.072798	0.023380
avg_word	-0.017643	-0.354335	1.000000	0.317892	0.025743	-0.051617	-0.001852
hashtags	-0.036279	0.016857	0.317892	1.000000	-0.012215	-0.045744	0.023881
exclamation_marks	-0.051845	0.003394	0.025743	-0.012215	1.000000	0.008317	0.002363
question_marks	0.045673	0.072798	-0.051617	-0.045744	0.008317	1.000000	0.025171
upper	0.007141	0.023380	-0.001852	0.023881	0.002363	0.025171	1.000000

Data Cleaning

Data cleaning done:

- Standardisation of tweets
    - Make all words lowercase so that model is not case-sensitive (e.g., "Hello" == "hello")
    - Remove punctuation (although certain punctuation structure might be a predictor for negative sentiments, this is extremely hard to analyse especially since informal sentence structure is used in tweets)
- Noise reduction
    - Remove stop words such as "the", "is" and "and" which are common and have a neutral sentiment. This reduces the amount of noise in the model and the dimensionality of analysis.
    - Removal of other common words. For the same reason as above, the most common word was @user.
    - Removal of rarely used words (used only once in all of data set). These words add extra noise since there is not a great enough sample to effectively analyse them.
- Spelling correction (Optional)

    - People often make spellings on short-form tweets. A lexicon of words and TextBlob can be used to fix easy to make corrections. However, this can be unreliable because non-standard words and abbreviations of words might occur e.g., 'u' instead of 'you' would not be automatically corrected. I experimented with this approach, but it is too unreliable and, since language changes quickly on social media, it is untenable to make a long-term model from this.
- Suffix and prefix removal (Optional)
    - The lexicon of analysed words can be reduced by reducing words to their 'unembellished form'. For example, 'hating', 'hater', 'hated', 'hate' is all reduced to 'hate'. Two approaches to this exist. The first is by removing any instances of a suffix or prefix directly (this is the simpler approach). The other is to use lemmatization to find the root word or conjugate that belongs to a specific word. Both approaches have been tried here although it makes sense to only use one at a time.
- Tokenization
    - After all common, rare, and insignificant words have been removed from tweets, tokenization can be used to create a list of the words that exist in the tweet. This removes duplicates but comes with the advantage that we can later vectorise each word, therefore the tweet, so it can be used as a feature for the model.

Appendix: Various Code Samples

```python
# Standardise column names
train_df.columns = list(map(lambda x: x.strip('_').lower(), list(train_df.columns)))
test_df.columns = list(map(lambda x: x.strip('_').lower(), list(test_df.columns)))

# Remove duplicate tweets
train_df.drop_duplicates(subset=['tweet'], keep='last', inplace=True)
test_df.drop_duplicates(subset=['tweet'], keep='last', inplace=True)

# Remove null tweets
train_df.dropna(subset=['tweet'], inplace=True)
test_df.dropna(subset=['tweet'], inplace=True)

# Remove unlabelled training data
train_df.dropna(subset=['label'], inplace=True)

# # Reindex to remove null or duplicate indexes
# train_df = train_df.reindex(np.arange(train_df.index.min(), train_df.index.max() + 1))
# test_df = test_df.reindex(np.arange(test_df.index.min(), test_df.index.max() + 1))
```

Name: Faris Chaudhry
Batch: LISUM25

```
# Character and word count
# train_df['char_count'] = train_df['tweet'].str.len() # Character count is redundant with word and average characters in word.
train_df['word_count'] = train_df['tweet'].apply(lambda x: len(str(x).split(" ")))

# Average word length
train_df['avg_word'] = train_df['tweet'].apply(lambda x: avg_word_length(x))

# Hashtags
train_df['hashtags'] = train_df['tweet'].apply(lambda x: len([x for x in x.split() if x.startswith('#')]))

# Exclamation marks
train_df['exclamation_marks'] = train_df['tweet'].str.count('!')

# Question Marks
train_df['question_marks'] = train_df['tweet'].str.count('\?')

# Uppercase words
train_df['upper'] = train_df['tweet'].apply(lambda x: len([x for x in x.split() if x.isupper()]))

train_df
```

Standardisation

```
# Make all words lowercase (no long case-sensitive)
train_df['tweet'] = train_df['tweet'].apply(lambda x: " ".join(x.lower() for x in x.split()))

# Removal of punctuation
train_df['tweet'] = train_df['tweet'].str.replace('[^\w\s]','')

# Removal of stopwords
stop = stopwords.words('english')
train_df['tweet'] = train_df['tweet'].apply(lambda x: " ".join(x for x in x.split() if x not in stop))
```

✓ 0.5s

Remove noise from common and rare words in tweets

```
# Common word removal (top 5 words)
common = list(pd.Series(' '.join(train_df['tweet']).split()).value_counts()[:5].index)
train_df['tweet'] = train_df['tweet'].apply(lambda x: " ".join(x for x in x.split() if x not in common))

# Rare word removal (all words with count of 1)
rare = pd.Series(' '.join(train_df['tweet']).split()).value_counts().to_frame()
rare = list(rare.loc[(rare['count'] <= 1)].index)
train_df['tweet'] = train_df['tweet'].apply(lambda x: " ".join(x for x in x.split() if x not in rare))
```

✓ 2m 54.9s

Tokenization

```
# Tokenizes each tweet
for index, row in train_df.iterrows():
    row['tweet'] = TextBlob(row['tweet']).words

train_df
```

✓ 6.1s

id	label	tweet	word_count	avg_word	hashtags	exclamation_marks	question_marks	upper
1	0	father selfish drags kids dysfunction. #run	21	4.556	1	0	0	0
2	0	thanks #lyft credit can't use cause offer whee...	22	5.316	3	0	0	0
3	0	bihday majesty	5	5.667	0	0	0	0
5	0	factsguide: society #motivation	8	8.000	1	0	0	0
6	0	huge fan fare big talking leave. chaos pay get...	21	5.053	1	0	0	0
...	...	...	...	...	...	...	...	...
31958	0	ate	6	12.600	0	0	1	0
31959	0	see nina turner trying wrap genuine hero like ...	25	4.652	2	0	0	0