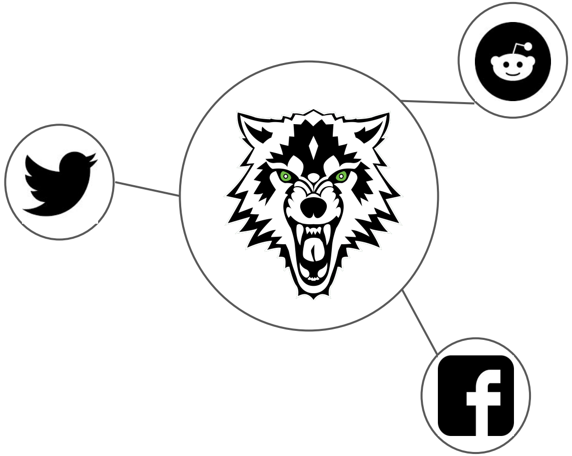
Automatic Abuse Detection on Social Media

#MeToo

#MeToo

#MeToo



#MeToo

#MeToo

#MeToo

Project by Abhishek Srivastava

Under the guidance of Dr Santosh Singh

Introduction

1. Within the past year, #MeToo has gained popularity on Facebook, Reddit, and Twitter.

2. Many other hashtags have come and gone:
   a. #YesAllWomen
   b. #WhyIStayed
   c. #ItsNotOkay

3. Automatic detection, classification and interpretation of personal abuse stories can help activist groups educate the public and advocate for social change in timely fashion.

Related Work

1. **Schrading et al (2015)** assembled the Reddit Domestic Abuse Dataset. They used multiple traditional classifiers e.g., Linear SVM, logistic regression, Naive Bayes, Random Forest, etc. Highest accuracy achieved was 92.0% using Linear SVM(C=1) with N-gram features.

2. **Karlekar et al (2018)** used the same dataset with different deep learning architectures (CNN, LSTM-RNN, CNN-LSTM) They achieved an accuracy of 95.8% with CNN-LSTM model.

Model	Accuracy
Schrading et al. (2015)	92.0%
2D-CNN	92.6%
LSTM-RNN	94.5%
CNN-LSTM	**95.8%**

Table 2: Accuracy results on abuse story detection.

Dataset

- ***redditAbuseOnlyNgrams*** This data contains a larger set of even data (1336 submissions per class), with no semantic roles or predicates. It has the variables:
  - XTrain: A list of submission title and text concatenated together, 90% training size (1202 per class).
  - XTest: A list of submission title and text concatenated together, 10% testing size (134 per class).
  - labelsTrain: A list of labels (abuse or non_abuse), 1 entry per submission.
  - labelsTest: A list of labels (abuse or non_abuse), 1 entry per submission.
  - subIdsTrain: A list of reddit submission ids, 1 entry per submission.
  - subIdsTest: A list of reddit submission ids, 1 entry per submission.

- ***redditAbuseUneven*** This data is an uneven set of data with 1336 *abuse* submissions and 17020 *non-abuse* submissions. It has the variables:
  - XTrain: A list of submission title, text, and comment data concatenated together, 85% training size.
  - XTest: A list of submission title, text, and comment data concatenated together, 15% testing size.
  - labelsTrain: A list of labels (abuse or non_abuse), 1 entry per submission.
  - labelsTest: A list of labels (abuse or non_abuse), 1 entry per submission.
  - subIdsTrain: A list of reddit submission ids, 1 entry per submission.
  - subIdsTest: A list of reddit submission ids, 1 entry per submission.

Data Source: Schrading et al (2015)

Dataset

- ***redditAbuseOnlyNgrams*** This data contains a larger set of even data (1336 submissions per class), with no semantic roles or predicates. It has the variables:
  - XTrain: A list of submission title and text concatenated together, 90% training size (1202 per class).
  - XTest: A list of submission title and text concatenated together, 10% testing size (134 per class).
  - labelsTrain: A list of labels (abuse or non_abuse), 1 entry per submission.
  - labelsTest: A list of labels (abuse or non_abuse), 1 entry per submission.
  - subIdsTrain: A list of reddit submission ids, 1 entry per submission.
  - subIdsTest: A list of reddit submission ids, 1 entry per submission.

- ***redditAbuseUneven*** This data is an uneven set of data with 1336 *abuse* submissions and 17020 *non-abuse* submissions. It has the variables:
  - XTrain: A list of submission title, text, and comment data concatenated together, 85% training size.
  - XTest: A list of submission title, text, and comment data concatenated together, 15% testing size.
  - labelsTrain: A list of labels (abuse or non_abuse), 1 entry per submission.
  - labelsTest: A list of labels (abuse or non_abuse), 1 entry per submission.
  - subIdsTrain: A list of reddit submission ids, 1 entry per submission.
  - subIdsTest: A list of reddit submission ids, 1 entry per submission.

Schrading et al

Karlekar at al

Data Source: Schrading et al (2015)

Reference Paper

1. **Karlekar et al (2018)** : #MeToo: Neural Detection and Explanation of Language in Personal Abuse Stories

Models	
CNN	1. For each input, an embedding and a convolutional layer is applied, followed by a max-pooling layer. 2. No pre-trained word embeddings were used. 3. Filter sizes of [3, 4, 5] with 128 filters per filter were used. 4. The convolution features are then passed to a softmax layer, which outputs probabilities over two classes.
LSTM-RNN	1. LSTM-RNN with 128 hidden units. 2. Embedding layer followed by two LSTM hidden layers. 3. The final state is fed to a fully-connected layer and then a softmax layer, which gives the final output probabilities.
CNN-LSTM	1. The RNN was laid on top of CNN model

Proposed Tasks

1. Implement the Reference Paper

Karlekar et al (2018) : #MeToo: Neural Detection and Explanation of Language in Personal Abuse Stories

2. Evaluate the model with pre-trained embeddings *(Optional)*

3. Use character level features in CNN *(Optional)*

Implementation

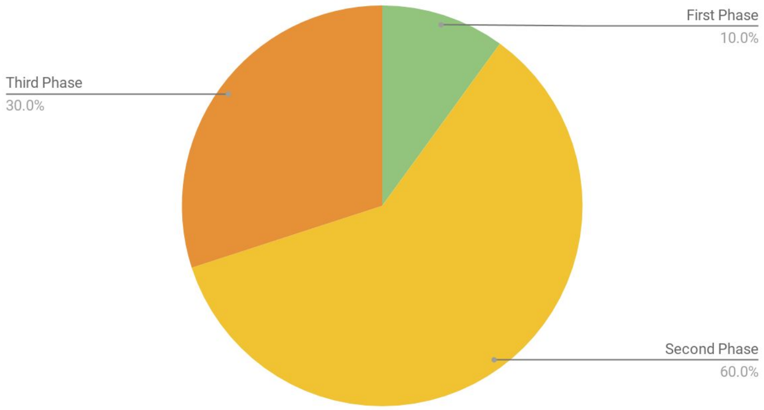
Libraries: Keras, Tensorflow

1. **First phase:** Pre-processing of Dataset
   a. **Load dataset** (Data already split in training/test set)
   b. **Visualise and Prepare Data**

2. **Second phase:** Implementation of Reference Paper
   a. **CNN model**
   b. **RNN - LSTM model**
   c. **CNN - LSTM model**

3. **Third phase:** Use pre-trained word embeddings
   a. **Glove** or **word2vec** embeddings
      i. Pre-trained
      ii. Trained on abuse dataset

Effort Estimation



Data Pre-Processing

x

In [116]:
```
#Function to tokenize the data and add a column

def tokenizer(text):
    text = clean_text(text)
    tokens = [word_tokenize(sent) for sent in sent_tokenize(text)]
    tokens = list(reduce(lambda x,y: x+y, tokens))
    tokens = list(filter(lambda token: token not in (stop + list(punctu
    return tokens
```

Data Pre-Processing

```
In [114]: #Function to remove Non-Ascii characters
          #Source: https://ahmedbesbes.com/how-to-mine-newsfeed-data-and-extract-

          def _removeNonAscii(s):
              return "".join(i for i in s if ord(i)<128)
```

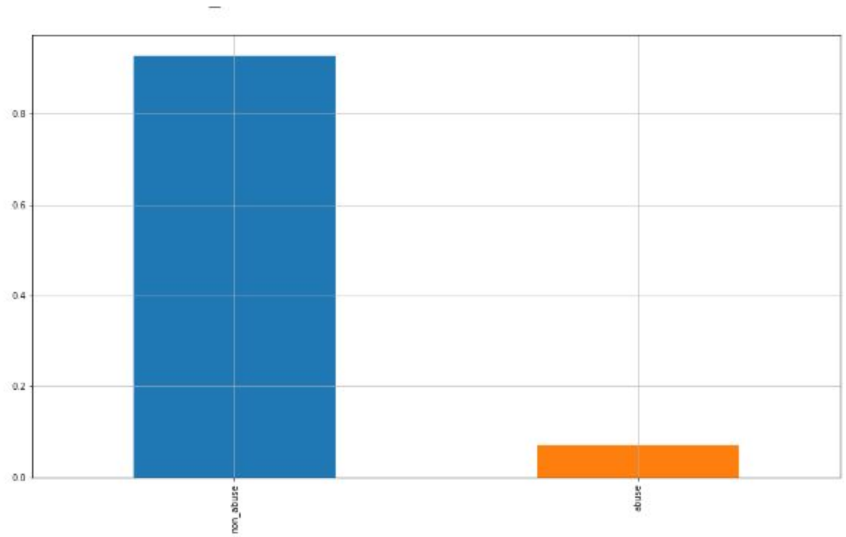
```
In [115]: #Function to clean text
          #Source: https://ahmedbesbes.com/how-to-mine-newsfeed-data-and-extract-

          def clean_text(text):
              text = text.lower()
              text = re.sub(r"what's", "what is ", text)
              text = text.replace('(ap)', '')
              text = re.sub(r"\'s", " is ", text)
              text = re.sub(r"\'ve", " have ", text)
              text = re.sub(r"can't", "cannot ", text)
              text = re.sub(r"n't", " not ", text)
              text = re.sub(r"i'm", "i am ", text)
              text = re.sub(r"\'re", " are ", text)
              text = re.sub(r"\'d", " would ", text)
              text = re.sub(r"\'ll", " will ", text)
              text = re.sub(r'\W+', ' ', text)
              text = re.sub(r'\s+', ' ', text)
              text = re.sub(r"\\", "", text)
              text = re.sub(r"\'", "", text)
              text = re.sub(r"\"", "", text)
              text = re.sub('[^a-zA-Z ?!]+', '', text)
              text = _removeNonAscii(text)
              text = text.strip()
              return text
```

x

Data Visualisation

1. Data Distribution

x



Data Visualisation

1. Most Common Words in Both Categories

x

Most Common Words

In [25]:
```
def keywords(label):
    tokens = data[data['label'] == label]['tokens']
    alltokens = []
    for token_list in tokens:
        alltokens += token_list
    counter = Counter(alltokens)
    return counter.most_common(20)
```

In [26]:
```
for label in set(data['label']):
    print('label :', label)
    print('top 10 keywords:', keywords(label))
    print('---')
```

```
label : abuse
top 10 keywords: [('would', 3018), ('like', 2711), ('know', 2296), ('g
et', 2231), ('abuse', 1855), ('time', 1852), ('feel', 1790), ('help',
1629), ('one', 1625), ('really', 1553), ('want', 1495), ('things', 145
5), ('people', 1401), ('even', 1375), ('think', 1330), ('could', 129
0), ('never', 1181), ('life', 1170), ('years', 1131), ('going', 1129)]
---
label : non_abuse
top 10 keywords: [('like', 52329), ('would', 40790), ('get', 37412),
('really', 30867), ('know', 29390), ('time', 29128), ('people', 2727
4), ('one', 27208), ('think', 22775), ('feel', 22098), ('want', 2182
0), ('go', 21760), ('good', 20596), ('going', 18402), ('something', 17
866), ('much', 17406), ('things', 17003), ('even', 16842), ('work', 16
807), ('could', 16342)]
---
```

Data Visualisation

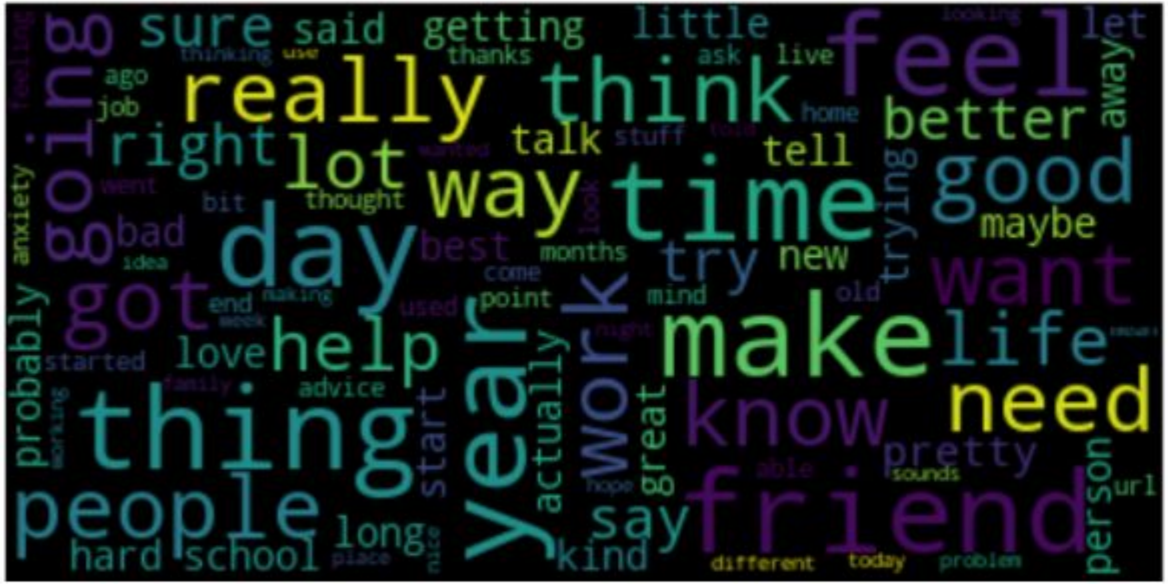
1. **WordCloud - Important Words using TF-IDF**

x

Data Visualisation

1. WordCloud - Least Important Words

x



Data Visualisation

1. Finding Topics using NMF

In [33]:
```
from sklearn.decomposition import NMF
```

In [34]:
```
nmf = NMF(n_components=40, random_state=1, alpha=.1, l1_ratio=.5, init=
```

x

In [40]:
```
feature_names = vectorizer.get_feature_names()
no_top_words = 5

for topic_idx, topic in enumerate(nmf.components_[:2]):
    print("Topic %d:"% (topic_idx))
    print(" | ".join([feature_names[i]
                    for i in topic.argsort()[:-no_top_words - 1:-1]]))
```

```
Topic 0:
like | really | know | time | think
Topic 1:
anxiety | anxious | help | symptoms | doctor
```

Data Visualisation

1. **Finding Topics using NMF**

x

```
In [43]: topic_table(nmf, feature_names, no_top_words).head(5)
```

Out[43]:

	topic_0:	topic_1:	topic_2:	topic_3:	topic_4:	topic_5:	topic_6:	topic_7:	topic_8:
0	like	anxiety	flair	url	job	mother	mom	anger	college
1	really	anxious	flair post	url url	work	sister	dad	angry	degree
2	know	help	add flair	picture	jobs	parents	house	rage	university
3	time	symptoms	added add	subredditlink comments	company	father	tell	control	schoo
4	think	doctor	bot tspq	selfie	working	brother	told	anger management	yea

Data Visualisation

1. Dividing topics as per abuse or non-abuse and finding intersection of topics

x

```
In [54]: """
             showdocs(df, topics, nshow=5) is a function that gathers a number o
             documents from a set of topics as a dataframe.
         """
         def showdocs(df, topics, nshow=5):
             idx = df.topic == topics[0]
             for i in range(1, len(topics)):
                 idx = idx | (df.topic == topics[i])
             return df[idx].groupby('topic').head(nshow).sort_values('topic')
```

```
In [55]: abuse = [0, 38, 41,  5, 43, 45,  3, 48, 29, 47]
         non_abuse = [0,  1,  4,  8,  3, 16, 11,  7, 19, 18]
```

```
In [64]: showdocs(df_abuse, [37,5])['body']

Out[64]: 10     sexually abused older brothers turns dad sexua...
         511    move forward wall text incoming last night mot...
         582    ai child sharon jones dap kings would run rais...
         618    first overnight babysitting job mother told fe...
         754    abuser died sexually molested mother father so...
         2      sucks man sometimes ex girlfriend used beat be...
         118    deal anymore point feel thing good abused smal...
         230    powerful south african psa bringing public apa...
         265    abuse equal abuse equal others rant ptsd somet...
         271    think need help fifteen father started asking ...
         Name: body, dtype: object
```

CNN Model

In [174]:
```
def create_conv_model():
    model_conv = Sequential()
    filters = 128
    model_conv.add(Embedding(vocabulary_size, 100, input_length=6002))
    model_conv.add(Dropout(0.2))
    model_conv.add(Conv1D(filters, 3, activation='relu'))
    model_conv.add(MaxPooling1D(pool_size=(3)))
    model_conv.add(Flatten())
    model_conv.add(Dense(1, activation='softmax'))
    model_conv.compile(loss='binary_crossentropy', optimizer='adam',
    return model_conv
```

x

Results

1. A model is designed that can classify stories as Abuse or Non Abuse
2. Data was visualised and techniques such as TF-IDF were employed to get important words
3. Hidden topics were decided using NMF and their relevance was shown to abuse and non-abuse stories

#ThankYou