SENTIMENTAL CLASSIFICATION OF MOVIE REVIEWS

1. Sukhad Joshi
2. Kulveen Kaur
3. Biswadip Bhattacharyya

INDEX

1. Dataset

This dataset, designed for the Kaggle competition available at [this link](https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews), combines information from Socher et al.'s sentiment analysis and Pang and Lee's original movie review corpus sourced from Rotten Tomatoes. Socher's team utilized crowd-sourcing to manually annotate sub-sentences with sentiment labels such as "Negative", "Little Negative", "Neutral", "Positive" and ""Little Positive". The data is divided into training and testing sets, with sentiment-labeled phrases provided in the train.tsv file and unlabeled phrases in test.tsv. Each sentence in the dataset must be associated with a sentiment label.

The following are the sentiment labels:

0 - negative

1 – little negative

2 - neutral

3 – little positive

4 – positive

2. Approach:

2.1. Reading Data from CSV file:

The primary function accepts two command-line arguments when run. The first argument specifies the directory path containing the train and test files, while the second argument represents the sample size. Within this function, the `processkaggle` function is invoked with these arguments. `processkaggle` performs initial processing tasks such as splitting the file into lines and subsequently calls the preprocessing function and feature set functions.

```
if __name__ == '__main__':
    if (len(sys.argv) != 3):
        print ('usage: classifyKaggle.py <corpus-dir> <limit>')
        sys.exit(0)
    processkaggle(sys.argv[1], sys.argv[2])
```

```
def processkaggle(dirPath,limitStr):
  # convert the limit argument from a string to an int
  limit = int(limitStr)

  os.chdir(dirPath)

  f = open('C:/Users/kulve/OneDrive/Documents/FinalProjectData (5)/FinalProjectData/kagglemoviereviews/corpus/train.tsv', 'r')
  # loop over lines in the file and use the first limit of them
  phrasedata = []
  for line in f:

    # ignore the first line starting with Phrase and read all lines
    if (not line.startswith('Phrase')):
      # remove final end of line character
      line = line.strip()
      # each line has 4 items separated by tabs
      # ignore th
      # e phrase and sentence ids, and keep the phrase and sentiment
      phrasedata.append(line.split('\t')[2:4])
```

2.2. Preprocessing and Filtering Data:

Both processed and unprocessed data was considered for all the experiments that were done.

2.2.1. Converting to Lowercase:

This line is used to covert it into lowercase and split into tokens

```
a = re.split(r'\s+', line.lower())
```

2.2.2. Removing punctuations:

Every token that is identified as punctuation during the tokenization process is eliminated from the list by replacing it with an empty string.

```
p = re.compile(r'[!#$%&()*+,"-./:;<=>?@[\]^_`{|}~]')
w = [p.sub("",i) for i in a]
```

2.2.3. Removing stop words:

The existing NLTK stopwords list has been augmented with additional words that could be classified as stopwords for

```
nltkstopwords = nltk.corpus.stopwords.words('english')
words_stop = [
    'could', 'would', 'might', 'must', 'need', 'sha', 'wo', 'y', "'s", "'d", "'ll",
    "'t", "'m", "'re", "'ve", "n't", "'i", 'not', 'no', 'can', 'don', 'nt',
    'actually', 'also', 'always', 'even', 'ever', 'just', 'really', 'still',
    'yet', 'however', 'nevertheless', 'furthermore', 'therefore', 'otherwise',
    'meanwhile', 'though', 'although', 'thus', 'hence', 'indeed', 'perhaps',
    'especially', 'specifically', 'usually', 'often', 'sometimes', 'certainly',
    'sometimes', 'typically', 'mostly', 'generally', 'about', 'above', 'across',
    'after', 'against', 'among', 'around', 'at', 'before', 'behind', 'below',
    'beneath', 'beside', 'between', 'beyond', 'during', 'inside', 'onto', 'outside',
    'through', 'under', 'upon', 'within', 'without'
]
stopwords = nltkstopwords + words_stop
```

```
y = []
for x in w:
    if x in stopwords:
        continue
    else:
        y.append(x)
l = " ".join(y)
return l
```

2.2.4. Filtering word tokens:

A distinct function called `filter_tokens2()` was developed to eliminate tokens from the list that had a length of less than 2 characters. This was necessary because certain words such as 'em' and 'nt' were identified, which were considered irrelevant in the context.

```python
def ft(t):
    a=[]
    for n in t[0]:
        if len(n)>2:
            a.append(n)
    return (a,t[1])
```

2.3. Generating Feature sets:

Different functions has been used to generate feature sets for both preprocessed and unprocessed data.

This is used for generating two lists of preprocessed and unprocessed tokens.

```python
withpreprocessing = []
withoutpreprocessing= []

for p in phraselist:


    tokens = nltk.word_tokenize(p[0])
    withoutpreprocessing.append((tokens, int(p[1])))


    p[0] = preprocessing(p[0])
    tokens = nltk.word_tokenize(p[0])
    withpreprocessing.append((tokens, int(p[1])))
```

This is used for generate Filtered list for Preprocessed tokens and list for unprocessed tokens:

```
withpreprocessing_filter=[]

for p in withpreprocessing:
    withpreprocessing_filter.append(ft(p))

filtered_tokens =[]
unfiltered_tokens = []
for (d,s) in  withpreprocessing_filter:
    for i in d:
        filtered_tokens.append(i)

for (d,s) in withoutpreprocessing:
    for i in d:
        unfiltered_tokens.append(i)
```

2.3.1. Bag of words:

```
def bw(a,i):
    a = nltk.FreqDist(a)
    wf = [w for (w,c) in a.most_common(i)]
    return wf
```

```
filtered_bow_features = bw(filtered_tokens,350)
unfiltered_bow_features = bw(unfiltered_tokens,350)
```

2.3.2. Unigram:

Unigram features are extracted from the documents or reviews, where each feature is represented with a label in the format "V_labelname". This process involves converting all words into features.

```
def uf(d,wf):
  df= set(d)
  f = {}
  for word in wf:
    f['V_%s'% word] = (word in df)
  return f
```

Unigram features are extracted for both filtered and unfiltered tokens.

```
filtered_unigram_features = [(uf(d,filtered_tokens),s) for (d,s) in withpreprocessing_filter]
unfiltered_unigram_features = [(uf(d,unfiltered_tokens),s) for (d,s) in withoutpreprocessing]
```

2.3.3. Bigram:

The `bigram_bow` function extracts significant bigram features from a list of words by applying frequency and chi-squared filters, while `bigram_features` extracts bigram features from a document using NLTK's `BigramCollocationFinder`. These functions are used for both unfiltered and filtered data to compare results.

```
def bigram_bow(wordlist,n):
  bigram_measure = nltk.collocations.BigramAssocMeasures()
  finder = BigramCollocationFinder.from_words(wordlist)
  finder.apply_freq_filter(2)
  b_features = finder.nbest(bigram_measure.chi_sq,4000)
  return b_features[:n]
```

```python
def bf(doc,word_features,bigram_feature):
    dw = set(doc)
    db = nltk.bigrams(doc)
    features = {}

    for word in word_features:
        features['V_{}'.format(word)] = (word in dw)

    for b in bigram_feature:
        features['B_{}_{}'.format(b[0],b[1])] = (b in db)

    return features
```

```python
filtered_bigram_features = [(bf(d,filtered_bow_features,bigram_bow(filtered_tokens,350)),s) for (d,s) in withpreprocessing_filter]
unfiltered_bigram_features = [(bf(d,unfiltered_bow_features,bigram_bow(unfiltered_tokens,350)),s) for (d,s) in withoutpreprocessing]
```

The above functions are used for extraction of bigram features for filtered and unfiltered data.

2.3.4. POS tagging:

The function extracts part-of-speech (POS) tagged features from documents by counting the occurrences of nouns, verbs, adjectives, and adverbs. This approach utilizes POS tagging information to capture the distribution of different word categories within the text.

```python
def pf(document, word_features):
    document_words = set(document)
    tagged_words = nltk.pos_tag(document)
    features = {}
    for word in word_features:
        features['contains({})'.format(word)] = (word in document_words)
    numNoun = 0
    numVerb = 0
    numAdj = 0
    numAdverb = 0
    for (word, tag) in tagged_words:
        if tag.startswith('N'): numNoun += 1
        if tag.startswith('V'): numVerb += 1
        if tag.startswith('J'): numAdj += 1
        if tag.startswith('R'): numAdverb += 1
    features['nouns'] = numNoun
    features['verbs'] = numVerb
    features['adjectives'] = numAdj
    features['adverbs'] = numAdverb
    return features
```

```python
filtered_pos_features = [(pf(d,filtered_bow_features),s) for (d,s) in withpreprocessing_filter]
unfiltered_pos_features = [(pf(d,unfiltered_bow_features),s) for (d,s) in withoutpreprocessing]
```

2.3.5. Sentiment Lexicon:

The program reads subjective words from a lexicon file sourced from the MPQA project led by Janice Wiebe and her team at the University of Pittsburgh. These subjective words are crucial for sentiment analysis as they help quantify the presence of positive and negative sentiment in each sentence or document. Each word in the lexicon is associated with intensity and polarity information. Weak subjective words encompass both positive and negative sentiments, while strong subjective words are counted separately for positive and negative sentiments. The program keeps track of positive and negative counts for each document to analyze sentiment polarity.

```python
def slf(document, word_features, SL):
    document_words = set(document)
    features = {}
    for word in word_features:
        features['V_{}'.format(word)] = (word in document_words)
    # count variables for the 4 classes of subjectivity
    weakPos = 0
    strongPos = 0
    weakNeg = 0
    strongNeg = 0
    for word in document_words:
        if word in SL:
            strength, posTag, isStemmed, polarity = SL[word]
            if strength == 'weaksubj' and polarity == 'positive':
                weakPos += 1
            if strength == 'strongsubj' and polarity == 'positive':
                strongPos += 1
            if strength == 'weaksubj' and polarity == 'negative':
                weakNeg += 1
            if strength == 'strongsubj' and polarity == 'negative':
                strongNeg += 1
            features['positivecount'] = weakPos + (2 * strongPos)
            features['negativecount'] = weakNeg + (2 * strongNeg)

    if 'positivecount' not in features:
      features['positivecount'] = 0
    if 'negativecount' not in features:
      features['negativecount'] = 0

    return features
```

For both filtered and unfiltered tokens, negated features were extracted.

```python
filtered_sl_features = [(slf(d, filtered_bow_features, SL), c) for (d, c) in withpreprocessing_filter]
unfiltered_sl_features = [(slf(d, unfiltered_bow_features, SL), c) for (d, c) in withoutpreprocessing]
```

2.3.6. LIWC features:

The LIWC (Linguistic Inquiry and Word Count) program is designed for text analysis, categorizing words into various linguistic, psychological, and topical categories to capture social, cognitive, and affective processes. In this context, the sentiment_read_LIWC_pos_neg_words.py package provides lists of words categorized into positive and negative emotion classes. The poslist and neglist parameters are initialized from the SL Lexicon tiff file, which contains lists of positive, neutral, and negative words. Using these lists, LIWC features are extracted for positive and negative words. This function is applied to both filtered and unfiltered data to extract features for sentiment classification.

```python
def liwc(doc,word_features,poslist,neglist):
    doc_words = set(doc)
    features= {}

    for word in word_features:
        features['contains({})'.format(word)] = (word in doc_words)

    pos = 0
    neg = 0
    for word in doc_words:
        if sentiment_read_LIWC_pos_neg_words.isPresent(word,poslist):
            pos+=1
        elif sentiment_read_LIWC_pos_neg_words.isPresent(word,neglist):
            neg+=1
        features ['positivecount'] = pos
        features ['negativecount'] = neg


    if 'positivecount' not in features:
        features['positivecount'] = 0
    if 'negativecount' not in features:
        features['negativecount'] = 0

    return features
```

This two functions are used to extract LIWC features for filtered and unfiltered data.

```python
filtered_liwc_features = [(liwc(d, filtered_bow_features, poslist,neglist), c) for (d, c) in withpreprocessing_filter]
unfiltered_liwc_features = [(liwc(d, unfiltered_bow_features, poslist,neglist), c) for (d, c) in withoutpreprocessing]
```

2.3.7. Combination of LIWC and SL:

The combined feature extraction method integrates both LIWC (Linguistic Inquiry and Word Count) and SL (subjectivity lexicon) features. In this approach, strong positive and strong negative features are counted twice, as they are detected both by LIWC and SL. However, weak positive and weak negative features are counted solely through the SL feature method. This combined approach leverages the strengths of both LIWC and SL to enhance the sentiment classification process.

```
def combo(doc,word_features,SL,poslist,neglist):
    doc_words = set(doc)
    features={}

    for word in word_features:
        features['contains({})'.format(word)] = (word in doc_words )

    weakPos = 0
    strongPos = 0
    weakNeg = 0
    strongNeg = 0
    for word in doc_words:
        if sentiment_read_LIWC_pos_neg_words.isPresent(word,poslist):
            strongPos +=1
        elif sentiment_read_LIWC_pos_neg_words.isPresent(word,neglist):
            strongNeg +=1
        elif word in SL:
            strength, posTag, isStemmed, polarity = SL[word]
            if strength == 'weaksubj' and polarity == 'positive':
                weakPos += 1
            if strength == 'strongsubj' and polarity == 'positive':
                strongPos += 1
            if strength == 'weaksubj' and polarity == 'negative':
                weakNeg += 1
            if strength == 'strongsubj' and polarity == 'negative':
                strongNeg += 1
        features['positivecount'] = weakPos + (2 * strongPos)
        features['negativecount'] = weakNeg + (2 * strongNeg)

    if 'positivecount' not in features:
        features['positivecount'] = 0
    if 'negativecount' not in features:
        features['negativecount'] = 0

    return features
```

For both filtered and unfiltered tokens, features were generated.

```
filtered_combo_features =  [(combo(d, filtered_bow_features,SL, poslist,neglist), c) for (d, c) in withpreprocessing_filter]
unfiltered_combo_features = [(combo(d, unfiltered_bow_features,SL, poslist,neglist), c) for (d, c) in withoutpreprocessing]
```

2.4. Saving Feature sets to CSV files:

All the generated feature sets have been saved into CSV files for future use as training sets with other classifiers or in separate Python notebooks. This approach ensures that the feature sets are readily available for analysis and modeling, even if computational constraints prevent immediate use in another Python script.

```
def save(features, path):
    f = open(path, 'w')
    featurenames = features[0][0].keys()
    fnameline = ''
    for fname in featurenames:
        fname = fname.replace(',','COM')
        fname = fname.replace("'","SQ")
        fname = fname.replace('"','DQ')
        fnameline += fname + ','
    fnameline += 'Level'
    f.write(fnameline)
    f.write('\n')
    for fset in features:
        featureline = ''
        for key in featurenames:
            # Check if the key exists in the feature set
            if key in fset[0]:
                featureline += str(fset[0][key]) + ','
            else:
                featureline += 'NA,'  # If the key does not exist, write 'NA' instead
        if fset[1] == 0:
          featureline += str("Less Negitive")
        elif fset[1] == 1:
          featureline += str("Strong negitive")
        elif fset[1] == 2:
          featureline += str("Neutral")
        elif fset[1] == 3:
          featureline += str("Strongly positive")
        elif fset[1] == 4:
          featureline += str("Less positive")
        f.write(featureline)
        f.write('\n')
    f.close()
```

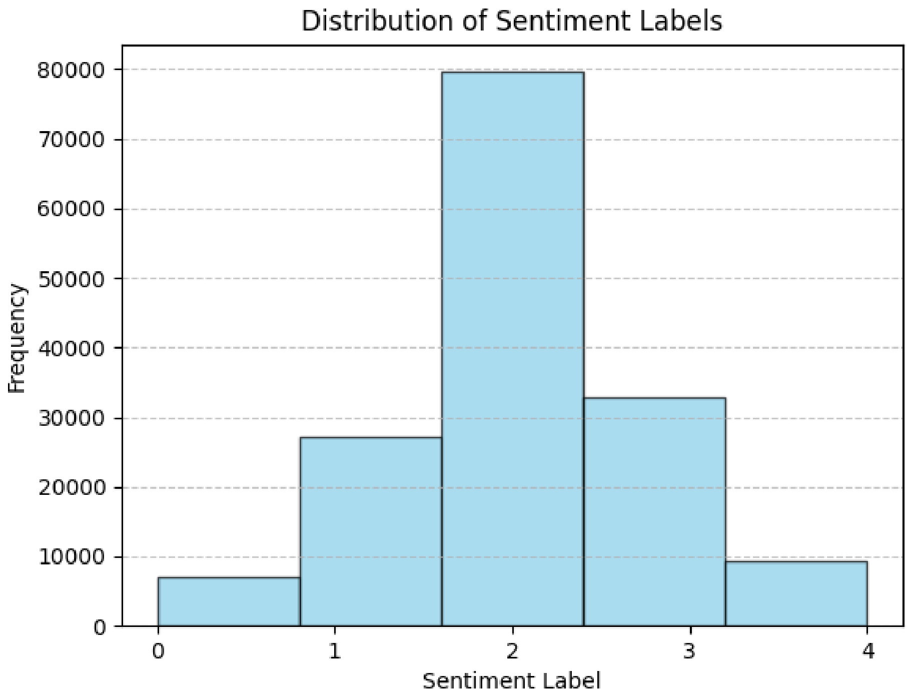
To save features following lines were used

```
save(filtered_unigram_features,'filtered_unigram.csv')
save(unfiltered_unigram_features,'unfiltered_unigram.csv')

save(filtered_bigram_features,'filtered_bigram.csv')
save(unfiltered_bigram_features,'unfiltered_bigram.csv')

save(filtered_pos_features,'filtered_pos.csv')
save(unfiltered_pos_features,'unfiltered_pos.csv')


save(filtered_sl_features,'filtered_sl.csv')
save(unfiltered_sl_features,'unfiltered_sl.csv')

save(filtered_liwc_features,'filtered_liwc.csv')
save(unfiltered_liwc_features,'unfiltered_liwc.csv')

save(filtered_combo_features,'filtered_combo.csv')
save(unfiltered_combo_features,'unfiltered_combo.csv')
```

2.5. Data Visualization:

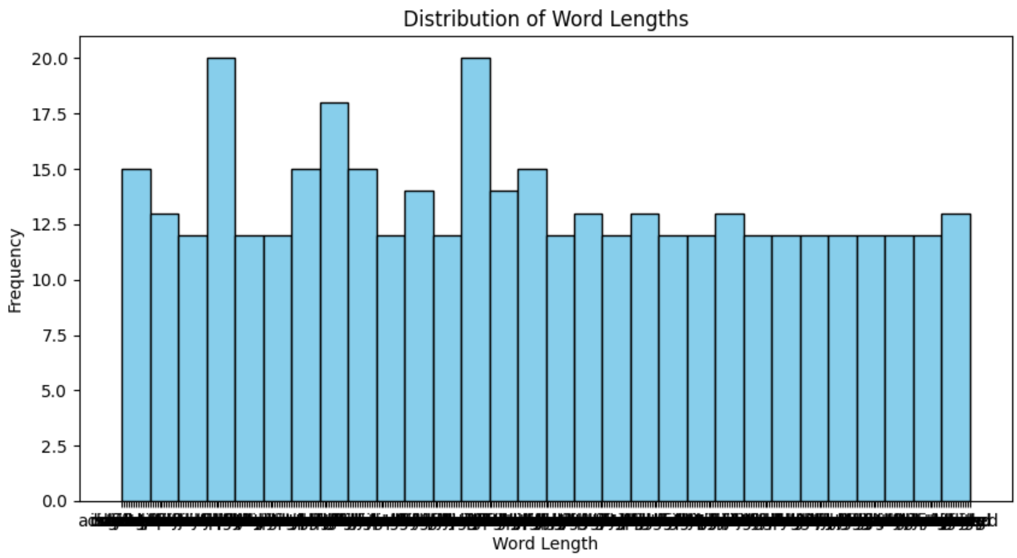
2.5.1 Sentiment Distribution Histogram

This histogram visualises the distribution of sentiment labels within the dataset. It provides insights into how frequently each sentiment label occurs and the overall composition of sentiment within the dataset.The height of each bar represents the frequency or count of occurrences for a particular sentiment label. Higher bars indicate a higher frequency of that sentiment label within the dataset.By observing the distribution of bars across different sentiment labels, we can discern the overall sentiment composition of the dataset. For example, we can see that sentiment 2 means neutral sentiment has highest frequency in the dataset .

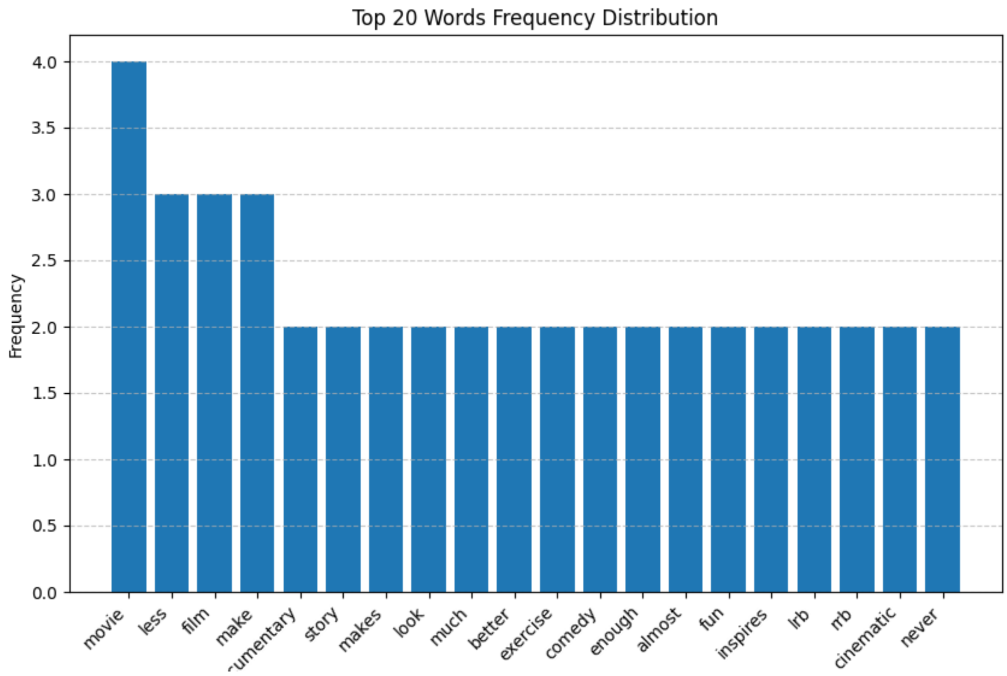


2.5.2 Word Frequency Distribution Histogram

The generated plot visualises the frequency distribution of words within the dataset. It provides insights into how frequently each word occurs and the overall composition of words in the text data. Here's what the graph tells us:The plot displays the most common words in the dataset, with the x-axis representing the words and the y-axis representing their frequencies. The higher the bar, the more frequently the word appears in the dataset.

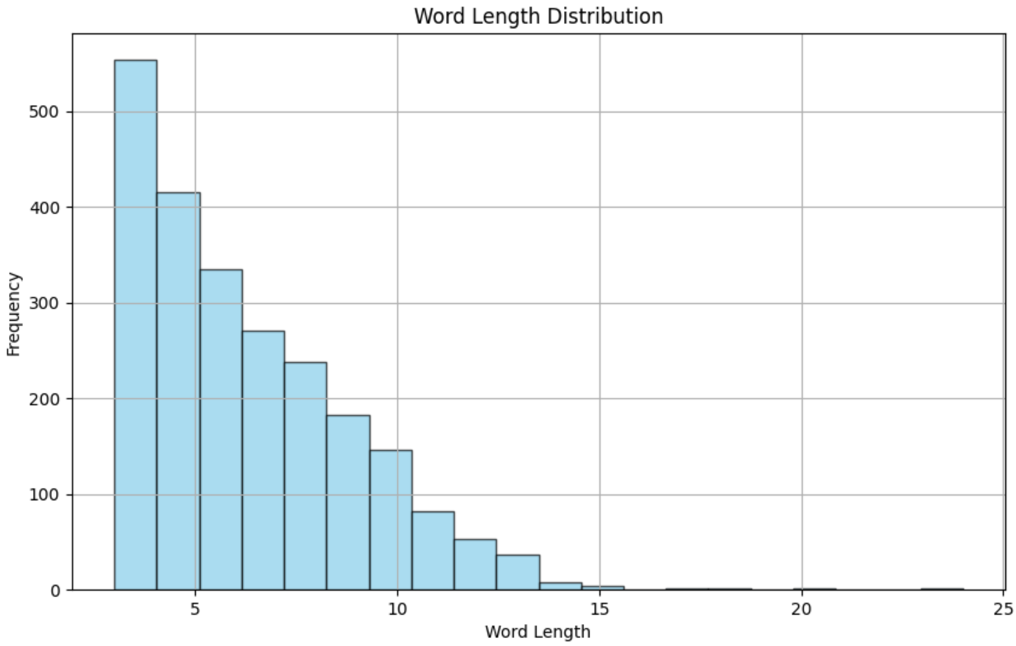


To get more visualisation and understanding of the word distribution we plot the most common 20 words .The plot displays the top 20 most common words in the dataset, with the x-axis representing the words and the y-axis representing their frequencies. Each bar's height indicates the frequency of occurrence of the corresponding word in the dataset.Words with higher bars are more significant in the dataset, as they occur more frequently. Hence we can see that ***movie*** word has the highest frequency while ***less , film and make*** have less than that .



2.5.3 Word length Histogram

The generated histogram visualises the distribution of word lengths within the dataset. It displays the frequency of different word lengths, with the x-axis representing the word lengths and the y-axis representing their frequencies. Each bar's height indicates the frequency of words with the corresponding length in the dataset.By observing the distribution of bars, we can identify the most common word lengths within the dataset. We can see that word length 2 -4 has occurred most in the phrases .



2.5.4 Word Cloud

This visualisation technique is used to represent text data in which the size of each word indicates its frequency or importance within the text.The size of each word in the word cloud corresponds to its frequency in the input text. Words that appear more frequently in the text will be displayed with larger font sizes, while less frequent words will appear smaller or may not be displayed at all.For example with this word cloud we can say that the words like ***movie , make less , film , offer*** are most frequently occurred as described with frequency graph .

2.6. Experiments :

The cross-validation process involved using various feature sets generated from the data. Each feature set was evaluated using 5-fold cross-validation. The evaluation metrics used were accuracy, precision, recall, and F1-score.For both unfiltered and filtered data, the Combined SL-LIWC feature set consistently resulted in the highest average scores across all evaluation measures.

The cross-validation process involved running the data through the functions provided in the `crossval.py` package. These functions implement cross-validation and evaluation measures, computing accuracy scores for each batch of data (each batch consisting of 31200 entries). After running the data through all batches (a total of 5 batches), the mean accuracy, precision, recall, and F1-scores were calculated collectively.

```
def cross_validation_PRF(num_folds, featuresets, labels):
    subset_size = int(len(featuresets)/num_folds)
    print('Each fold size:', subset_size)
    # for the number of labels - start the totals lists with zeroes
    num_labels = len(labels)
    total_precision_list = [0] * num_labels
    total_recall_list = [0] * num_labels
    total_F1_list = [0] * num_labels

    # iterate over the folds
    for i in range(num_folds):
        test_this_round = featuresets[(i*subset_size):][:subset_size]
        train_this_round = featuresets[:(i*subset_size)] + featuresets[((i+1)*subset_size):]
        # train using train_this_round
        classifier = nltk.NaiveBayesClassifier.train(train_this_round)
        # evaluate against test_this_round to produce the gold and predicted labels
        goldlist = []
        predictedlist = []
        for (features, label) in test_this_round:
            goldlist.append(label)
            predictedlist.append(classifier.classify(features))

        # computes evaluation measures for this fold and
        #   returns list of measures for each label
        print('Fold', i)
        (precision_list, recall_list, F1_list) \
                  = eval_measures(goldlist, predictedlist, labels)
        # take off triple string to print precision, recall and F1 for each fold

        #calculating accuracy
        accuracy_list= []
        accuracy_this_round = nltk.classify.accuracy(classifier,test_this_round)
        accuracy_list.append(accuracy_this_round)
```

```
        print('\tPrecision\tRecall\t\tF1')
        # print measures for each label
        for i, lab in enumerate(labels):
            print(lab, '\t', "{:10.3f}".format(precision_list[i]), \
              "{:10.3f}".format(recall_list[i]), "{:10.3f}".format(F1_list[i]))

        # for each label add to the sums in the total lists
        for i in range(num_labels):
            # for each label, add the 3 measures to the 3 lists of totals
            total_precision_list[i] += precision_list[i]
            total_recall_list[i] += recall_list[i]
            total_F1_list[i] += F1_list[i]

    # find precision, recall and F measure averaged over all rounds for all labels
    # compute averages from the totals lists
    precision_list = [tot/num_folds for tot in total_precision_list]
    recall_list = [tot/num_folds for tot in total_recall_list]
    F1_list = [tot/num_folds for tot in total_F1_list]

    print('\nAverage Accuracy : ', sum(accuracy_list)/num_folds)
    # the evaluation measures in a table with one row per label
    print('\nAverage Precision\tRecall\t\tF1 \tPer Label')
    # print measures for each label
    for i, lab in enumerate(labels):
        print(lab, '\t', "{:10.3f}".format(precision_list[i]), \
          "{:10.3f}".format(recall_list[i]), "{:10.3f}".format(F1_list[i]))

    # print macro average over all labels - treats each label equally
    print('\nMacro Average Precision\tRecall\t\tF1 \tOver All Labels')
    print('\t', "{:10.3f}".format(sum(precision_list)/num_labels), \
          "{:10.3f}".format(sum(recall_list)/num_labels), \
          "{:10.3f}".format(sum(F1_list)/num_labels))
```

```
    # first initialize a dictionary for label counts and then count them
    label_counts = {}
    for lab in labels:
        label_counts[lab] = 0
    # count the labels
    for (doc, lab) in featuresets:
        label_counts[lab] += 1
    # make weights compared to the number of documents in featuresets
    num_docs = len(featuresets)
    label_weights = [(label_counts[lab] / num_docs) for lab in labels]
    print('\nLabel Counts', label_counts)
    #print('Label weights', label_weights)
    # print macro average over all labels
    print('Micro Average Precision\tRecall\t\tF1 \tOver All Labels')
    precision = sum([a * b for a,b in zip(precision_list, label_weights)])
    recall = sum([a * b for a,b in zip(recall_list, label_weights)])
    F1 = sum([a * b for a,b in zip(F1_list, label_weights)])
    print( '\t', "{:10.3f}".format(precision), \
        "{:10.3f}".format(recall), "{:10.3f}".format(F1))
```

```
def eval_measures(gold, predicted, labels):

    # these lists have values for each label
    recall_list = []
    precision_list = []
    F1_list = []

    for lab in labels:
        # for each label, compare gold and predicted lists and compute values
        TP = FP = FN = TN = 0
        for i, val in enumerate(gold):
            if val == lab and predicted[i] == lab:  TP += 1
            if val == lab and predicted[i] != lab:  FN += 1
            if val != lab and predicted[i] == lab:  FP += 1
            if val != lab and predicted[i] != lab:  TN += 1
        # use these to compute recall, precision, F1
        # for small numbers, guard against dividing by zero in computing measures
        if (TP == 0) or (FP == 0) or (FN == 0):
          recall_list.append (0)
          precision_list.append (0)
          F1_list.append(0)
        else:
          recall = TP / (TP + FP)
          precision = TP / (TP + FN)
          recall_list.append(recall)
          precision_list.append(precision)
          F1_list.append( 2 * (recall * precision) / (recall + precision))

    # the evaluation measures in a table with one row per label
    return (precision_list, recall_list, F1_list)

## function to read kaggle training file, train and test a classifier
def processkaggle(dirPath,limitStr):
  # convert the limit argument from a string to an int
  limit = int(limitStr)
```

```
def processkaggle(dirPath,limitStr):
  # convert the limit argument from a string to an int
  limit = int(limitStr)

  os.chdir(dirPath)

  f = open('./train.tsv', 'r')
  # loop over lines in the file and use the first limit of them
  phrasedata = []
  for line in f:
    # ignore the first line starting with Phrase and read all lines
    if (not line.startswith('Phrase')):
      # remove final end of line character
      line = line.strip()
      # each line has 4 items separated by tabs
      # ignore the phrase and sentence ids, and keep the phrase and sentiment
      phrasedata.append(line.split('\t')[2:4])

  # pick a random sample of length limit because of phrase overlapping sequences
  random.shuffle(phrasedata)
  phraselist = phrasedata[:limit]

  print('Read', len(phrasedata), 'phrases, using', len(phraselist), 'random phrases')

  # create list of phrase documents as (list of words, label)
  phrasedocs = []
  # add all the phrases

  # each phrase has a list of tokens and the sentiment label (from 0 to 4)
  ### bin to only 3 categories for better performance
  for phrase in phraselist:
    tokens = nltk.word_tokenize(phrase[0])
    phrasedocs.append((tokens, int(phrase[1])))
```

```
docs = []
for phrase in phrasedocs:
  lowerphrase = ([w.lower() for w in phrase[0]], phrase[1])
  docs.append (lowerphrase)
# print a few
for phrase in docs[:10]:
  print (phrase)

# continue as usual to get all words and create word features
all_words_list = [word for (sent,cat) in docs for word in sent]
all_words = nltk.FreqDist(all_words_list)
print(len(all_words))

# get the 1500 most frequently appearing keywords in the corpus
word_items = all_words.most_common(1500)
word_features = [word for (word,count) in word_items]

# feature sets from a feature definition function
featuresets = [(document_features(d, word_features), c) for (d, c) in docs]

# train classifier and show performance in cross-validation
# make a list of labels
label_list = [c for (d,c) in docs]
labels = list(set(label_list))    # gets only unique labels
num_folds = 5
cross_validation_PRF(num_folds, featuresets, labels)
```

2.6.1 Cross Validation on featured Sets

Unigram

- Filtered

```
 Unigram filtered :
Each fold size: 100
Fold 0
        Precision     Recall        F1
0           0.000      0.000     0.000
1           0.133      0.667     0.222
2           0.909      0.625     0.741
3           0.278      0.294     0.286
4           0.000      0.000     0.000
Fold 1
        Precision     Recall        F1
0           0.000      0.000     0.000
1           0.000      0.000     0.000
2           0.976      0.441     0.607
3           0.143      0.429     0.214
4           0.000      0.000     0.000
Fold 2
        Precision     Recall        F1
0           0.000      0.000     0.000
1           0.000      0.000     0.000
2           0.930      0.589     0.721
3           0.125      0.375     0.188
4           0.000      0.000     0.000
Fold 3
        Precision     Recall        F1
0           0.000      0.000     0.000
1           0.053      0.111     0.071
2           0.925      0.468     0.622
3           0.133      0.364     0.195
4           0.000      0.000     0.000
Fold 4
        Precision     Recall        F1
0           0.000      0.000     0.000
1           0.105      0.667     0.182
2           0.900      0.536     0.672
3           0.048      0.077     0.059
4           0.000      0.000     0.000
```

```
Average Accuracy :  0.096

Average Precision       Recall          F1      Per Label
0               0.000      0.000      0.000
1               0.058      0.289      0.095
2               0.928      0.532      0.673
3               0.145      0.308      0.188
4               0.000      0.000      0.000

Macro Average Precision Recall          F1      Over All Labels
                0.226      0.226      0.191

Label Counts {0: 24, 1: 94, 2: 244, 3: 114, 4: 24}
Micro Average Precision Recall          F1      Over All Labels
                0.497      0.384      0.389
```

- Unfiltered

```
 Unigram Unfiltered :
Each fold size: 100
Fold 0
        Precision       Recall          F1
0             0.000      0.000      0.000
1             0.333      0.556      0.417
2             0.855      0.671      0.752
3             0.278      0.250      0.263
4             0.000      0.000      0.000
Fold 1
        Precision       Recall          F1
0             0.000      0.000      0.000
1             0.103      0.500      0.171
2             0.905      0.463      0.613
3             0.095      0.167      0.121
4             0.000      0.000      0.000
Fold 2
        Precision       Recall          F1
0             0.000      0.000      0.000
1             0.083      0.143      0.105
2             0.895      0.593      0.713
3             0.125      0.429      0.194
4             0.000      0.000      0.000
Fold 3
        Precision       Recall          F1
0             0.000      0.000      0.000
1             0.316      0.333      0.324
2             0.925      0.544      0.685
3             0.167      0.385      0.233
4             0.000      0.000      0.000
Fold 4
        Precision       Recall          F1
0             0.000      0.000      0.000
1             0.211      0.444      0.286
2             0.900      0.584      0.709
3             0.143      0.214      0.171
4             0.000      0.000      0.000
```

```
Average Accuracy :  0.10400000000000001

Average Precision       Recall          F1      Per Label
0               0.000      0.000      0.000
1               0.209      0.395      0.261
2               0.896      0.571      0.694
3               0.162      0.289      0.196
4               0.000      0.000      0.000

Macro Average Precision Recall          F1      Over All Labels
                0.253      0.251      0.230

Label Counts {0: 24, 1: 94, 2: 244, 3: 114, 4: 24}
Micro Average Precision Recall          F1      Over All Labels
                0.513      0.419      0.433
```

Bigram

- Filtered

```
 Bigram filtered :
Each fold size: 100
Fold 0
        Precision       Recall          F1
0               0.000      0.000      0.000
1               0.200      0.375      0.261
2               0.782      0.662      0.717
3               0.056      0.077      0.065
4               0.100      0.250      0.143
Fold 1
        Precision       Recall          F1
0               0.000      0.000      0.000
1               0.034      0.333      0.062
2               0.952      0.440      0.602
3               0.095      0.333      0.148
4               0.000      0.000      0.000
Fold 2
        Precision       Recall          F1
0               0.000      0.000      0.000
1               0.000      0.000      0.000
2               0.947      0.587      0.725
3               0.042      0.200      0.069
4               0.000      0.000      0.000
Fold 3
        Precision       Recall          F1
0               0.000      0.000      0.000
1               0.053      0.125      0.074
2               0.900      0.462      0.610
3               0.200      0.462      0.279
4               0.000      0.000      0.000
Fold 4
        Precision       Recall          F1
0               0.000      0.000      0.000
1               0.053      0.333      0.091
2               0.880      0.512      0.647
3               0.048      0.091      0.062
4               0.000      0.000      0.000
```

```
Average Accuracy :  0.092

Average Precision       Recall          F1      Per Label
0               0.000       0.000       0.000
1               0.068       0.233       0.098
2               0.892       0.532       0.660
3               0.088       0.233       0.125
4               0.020       0.050       0.029

Macro Average Precision Recall          F1      Over All Labels
                0.214       0.210       0.182

Label Counts {0: 24, 1: 94, 2: 244, 3: 114, 4: 24}
Micro Average Precision Recall          F1      Over All Labels
                0.469       0.359       0.370
```

- Unfiltered

```
 Bigram Unfiltered :
Each fold size: 100
Fold 0
        Precision       Recall          F1
0               0.000       0.000       0.000
1               0.333       0.500       0.400
2               0.855       0.691       0.764
3               0.278       0.312       0.294
4               0.000       0.000       0.000
Fold 1
        Precision       Recall          F1
0               0.250       0.333       0.286
1               0.241       0.500       0.326
2               0.881       0.487       0.627
3               0.095       0.400       0.154
4               0.000       0.000       0.000
Fold 2
        Precision       Recall          F1
0               0.000       0.000       0.000
1               0.167       0.167       0.167
2               0.860       0.605       0.710
3               0.042       0.250       0.071
4               0.000       0.000       0.000
Fold 3
        Precision       Recall          F1
0               0.111       0.250       0.154
1               0.368       0.333       0.350
2               0.825       0.541       0.653
3               0.200       0.500       0.286
4               0.000       0.000       0.000
Fold 4
        Precision       Recall          F1
0               0.000       0.000       0.000
1               0.158       0.250       0.194
2               0.860       0.566       0.683
3               0.095       0.200       0.129
4               0.000       0.000       0.000
```

```
Average Accuracy :  0.098

Average Precision       Recall          F1      Per Label
0               0.072       0.117       0.088
1               0.254       0.350       0.287
2               0.856       0.578       0.687
3               0.142       0.332       0.187
4               0.000       0.000       0.000

Macro Average Precision Recall          F1      Over All Labels
                0.265       0.275       0.250

Label Counts {0: 24, 1: 94, 2: 244, 3: 114, 4: 24}
Micro Average Precision Recall          F1      Over All Labels
                0.501       0.429       0.436
```

Pos Tagging

- Filtered

```
 Pos filtered :
Each fold size: 100
Fold 0
         Precision      Recall        F1
0           0.000     0.000     0.000
1           0.200     0.375     0.261
2           0.782     0.672     0.723
3           0.111     0.143     0.125
4           0.000     0.000     0.000
Fold 1
         Precision      Recall        F1
0           0.000     0.000     0.000
1           0.069     0.667     0.125
2           0.929     0.464     0.619
3           0.190     0.308     0.235
4           0.000     0.000     0.000
Fold 2
         Precision      Recall        F1
0           0.000     0.000     0.000
1           0.083     0.333     0.133
2           0.895     0.600     0.718
3           0.125     0.375     0.188
4           0.000     0.000     0.000
Fold 3
         Precision      Recall        F1
0           0.000     0.000     0.000
1           0.211     0.235     0.222
2           0.925     0.521     0.667
3           0.067     0.222     0.103
4           0.000     0.000     0.000
Fold 4
         Precision      Recall        F1
0           0.000     0.000     0.000
1           0.158     0.429     0.231
2           0.820     0.569     0.672
3           0.143     0.143     0.143
4           0.000     0.000     0.000
```

```
Average Accuracy :  0.094

Average Precision        Recall        F1      Per Label
0             0.000      0.000      0.000
1             0.144      0.408      0.194
2             0.870      0.565      0.680
3             0.127      0.238      0.159
4             0.000      0.000      0.000

Macro Average Precision Recall        F1      Over All Labels
              0.228      0.242      0.207

Label Counts {0: 24, 1: 94, 2: 244, 3: 114, 4: 24}
Micro Average Precision Recall        F1      Over All Labels
              0.481      0.407      0.404
```

- Unfiltered

```
 Pos Unfiltered :
Each fold size: 100
Fold 0
        Precision       Recall          F1
0             0.000      0.000      0.000
1             0.333      0.455      0.385
2             0.800      0.721      0.759
3             0.278      0.263      0.270
4             0.000      0.000      0.000
Fold 1
        Precision       Recall          F1
0             0.250      0.500      0.333
1             0.276      0.500      0.356
2             0.833      0.500      0.625
3             0.048      0.111      0.067
4             0.000      0.000      0.000
Fold 2
        Precision       Recall          F1
0             0.000      0.000      0.000
1             0.167      0.200      0.182
2             0.842      0.608      0.706
3             0.083      0.286      0.129
4             0.000      0.000      0.000
Fold 3
        Precision       Recall          F1
0             0.111      0.250      0.154
1             0.368      0.368      0.368
2             0.850      0.531      0.654
3             0.100      0.333      0.154
4             0.000      0.000      0.000
Fold 4
        Precision       Recall          F1
0             0.000      0.000      0.000
1             0.211      0.267      0.235
2             0.820      0.594      0.689
3             0.095      0.154      0.118
4             0.000      0.000      0.000
```

```
Average Accuracy :  0.096

Average Precision       Recall          F1      Per Label
0               0.072       0.150       0.097
1               0.271       0.358       0.305
2               0.829       0.591       0.686
3               0.121       0.229       0.147
4               0.000       0.000       0.000

Macro Average Precision Recall          F1      Over All Labels
                0.259       0.266       0.247

Label Counts {0: 24, 1: 94, 2: 244, 3: 114, 4: 24}
Micro Average Precision Recall          F1      Over All Labels
                0.487       0.415       0.431
```

SL Features

- Filtered

```
 SL filtered :
Each fold size: 100
Fold 0
        Precision       Recall          F1
0             0.000      0.000       0.000
1             0.333      0.357       0.345
2             0.727      0.690       0.708
3             0.056      0.059       0.057
4             0.100      0.167       0.125
Fold 1
        Precision       Recall          F1
0             0.000      0.000       0.000
1             0.034      0.333       0.062
2             0.929      0.464       0.619
3             0.143      0.231       0.176
4             0.000      0.000       0.000
Fold 2
        Precision       Recall          F1
0             0.000      0.000       0.000
1             0.000      0.000       0.000
2             0.930      0.609       0.736
3             0.167      0.571       0.258
4             0.000      0.000       0.000
Fold 3
        Precision       Recall          F1
0             0.000      0.000       0.000
1             0.053      0.100       0.069
2             0.900      0.474       0.621
3             0.167      0.417       0.238
4             0.000      0.000       0.000
Fold 4
        Precision       Recall          F1
0             0.000      0.000       0.000
1             0.105      0.400       0.167
2             0.880      0.537       0.667
3             0.143      0.231       0.176
4             0.000      0.000       0.000
```

```
Average Accuracy :  0.098

Average Precision       Recall          F1      Per Label
0             0.000      0.000       0.000
1             0.105      0.238       0.129
2             0.873      0.555       0.670
3             0.135      0.302       0.181
4             0.020      0.033       0.025

Macro Average Precision Recall          F1      Over All Labels
              0.227      0.226       0.201

Label Counts {0: 24, 1: 94, 2: 244, 3: 114, 4: 24}
Micro Average Precision Recall          F1      Over All Labels
              0.478      0.386       0.394
```

- Unfiltered

```
 SL Unfiltered :
Each fold size: 100
Fold 0
        Precision      Recall         F1
0           0.000       0.000      0.000
1           0.267       0.400      0.320
2           0.818       0.692      0.750
3           0.222       0.211      0.216
4           0.000       0.000      0.000
Fold 1
        Precision      Recall         F1
0           0.000       0.000      0.000
1           0.241       0.583      0.341
2           0.881       0.474      0.617
3           0.143       0.429      0.214
4           0.000       0.000      0.000
Fold 2
        Precision      Recall         F1
0           0.000       0.000      0.000
1           0.167       0.154      0.160
2           0.842       0.632      0.722
3           0.125       0.429      0.194
4           0.000       0.000      0.000
Fold 3
        Precision      Recall         F1
0           0.111       0.250      0.154
1           0.263       0.250      0.256
2           0.875       0.547      0.673
3           0.067       0.333      0.111
4           0.000       0.000      0.000
Fold 4
        Precision      Recall         F1
0           0.000       0.000      0.000
1           0.158       0.250      0.194
2           0.820       0.569      0.672
3           0.190       0.267      0.222
4           0.000       0.000      0.000
```

```
Average Accuracy :  0.096

Average Precision       Recall         F1      Per Label
0             0.022      0.050      0.031
1             0.219      0.327      0.254
2             0.847      0.583      0.687
3             0.149      0.334      0.191
4             0.000      0.000      0.000

Macro Average Precision Recall         F1      Over All Labels
              0.248      0.259      0.233

Label Counts {0: 24, 1: 94, 2: 244, 3: 114, 4: 24}
Micro Average Precision Recall         F1      Over All Labels
              0.490      0.424      0.428
```

LIWC Features

- Filtered

```
 LIWC filtered :
Each fold size: 100
Fold 0
        Precision      Recall         F1
0             0.000      0.000      0.000
1             0.200      0.429      0.273
2             0.764      0.667      0.712
3             0.167      0.176      0.171
4             0.100      0.143      0.118
Fold 1
        Precision      Recall         F1
0             0.000      0.000      0.000
1             0.000      0.000      0.000
2             0.952      0.465      0.625
3             0.190      0.333      0.242
4             0.000      0.000      0.000
Fold 2
        Precision      Recall         F1
0             0.000      0.000      0.000
1             0.083      0.250      0.125
2             0.930      0.609      0.736
3             0.125      0.429      0.194
4             0.000      0.000      0.000
Fold 3
        Precision      Recall         F1
0             0.000      0.000      0.000
1             0.105      0.200      0.138
2             0.875      0.479      0.619
3             0.167      0.357      0.227
4             0.000      0.000      0.000
Fold 4
        Precision      Recall         F1
0             0.000      0.000      0.000
1             0.053      0.250      0.087
2             0.860      0.537      0.662
3             0.095      0.125      0.108
4             0.000      0.000      0.000
```

```
Average Accuracy :  0.092

Average Precision       Recall          F1      Per Label
0             0.000      0.000      0.000
1             0.088      0.226      0.125
2             0.876      0.552      0.671
3             0.149      0.284      0.189
4             0.020      0.029      0.024

Macro Average Precision Recall          F1      Over All Labels
              0.227      0.218      0.201

Label Counts {0: 24, 1: 94, 2: 244, 3: 114, 4: 24}
Micro Average Precision Recall          F1      Over All Labels
              0.479      0.378      0.395
```

- Unfiltered

```
 LIWC Unfiltered :
Each fold size: 100
Fold 0
          Precision     Recall      F1
0             0.000      0.000   0.000
1             0.333      0.556   0.417
2             0.818      0.682   0.744
3             0.278      0.263   0.270
4             0.000      0.000   0.000
Fold 1
          Precision     Recall      F1
0             0.000      0.000   0.000
1             0.241      0.583   0.341
2             0.881      0.493   0.632
3             0.143      0.333   0.200
4             0.000      0.000   0.000
Fold 2
          Precision     Recall      F1
0             0.000      0.000   0.000
1             0.167      0.182   0.174
2             0.860      0.620   0.721
3             0.125      0.500   0.200
4             0.000      0.000   0.000
Fold 3
          Precision     Recall      F1
0             0.111      0.200   0.143
1             0.316      0.286   0.300
2             0.850      0.548   0.667
3             0.133      0.500   0.211
4             0.000      0.000   0.000
Fold 4
          Precision     Recall      F1
0             0.200      0.500   0.286
1             0.053      0.077   0.062
2             0.840      0.575   0.683
3             0.190      0.333   0.242
4             0.000      0.000   0.000
```

```
Average Accuracy :  0.096

Average Precision      Recall         F1      Per Label
0            0.062      0.140      0.086
1            0.222      0.337      0.259
2            0.850      0.584      0.689
3            0.174      0.386      0.225
4            0.000      0.000      0.000

Macro Average Precision Recall        F1      Over All Labels
             0.262      0.289      0.252

Label Counts {0: 24, 1: 94, 2: 244, 3: 114, 4: 24}
Micro Average Precision Recall        F1      Over All Labels
             0.499      0.443      0.440
```

<u>Combined SL and LIWC Features</u>

- Filtered

```
Combined SL LIWC filtered:
Each fold size: 100
Fold 0
          Precision      Recall        F1
0             0.000      0.000      0.000
1             0.267      0.333      0.296
2             0.764      0.700      0.730
3             0.167      0.176      0.171
4             0.100      0.143      0.118
Fold 1
          Precision      Recall        F1
0             0.000      0.000      0.000
1             0.000      0.000      0.000
2             0.929      0.470      0.624
3             0.238      0.312      0.270
4             0.000      0.000      0.000
Fold 2
          Precision      Recall        F1
0             0.000      0.000      0.000
1             0.000      0.000      0.000
2             0.912      0.598      0.722
3             0.125      0.429      0.194
4             0.000      0.000      0.000
Fold 3
          Precision      Recall        F1
0             0.000      0.000      0.000
1             0.053      0.091      0.067
2             0.875      0.507      0.642
3             0.233      0.412      0.298
4             0.000      0.000      0.000
Fold 4
          Precision      Recall        F1
0             0.000      0.000      0.000
1             0.105      0.400      0.167
2             0.880      0.550      0.677
3             0.143      0.200      0.167
4             0.000      0.000      0.000
```

```
Average Accuracy :  0.098

Average Precision       Recall          F1      Per Label
0             0.000     0.000     0.000
1             0.085     0.165     0.106
2             0.872     0.565     0.679
3             0.181     0.306     0.220
4             0.020     0.029     0.024

Macro Average Precision Recall          F1      Over All Labels
              0.232     0.213     0.206

Label Counts {0: 24, 1: 94, 2: 244, 3: 114, 4: 24}
Micro Average Precision Recall          F1      Over All Labels
              0.484     0.378     0.403
```

- Unfiltered

```
 Combined SL LIWC Unfiltered :
Each fold size: 100
Fold 0
        Precision      Recall       F1
0           0.000      0.000      0.000
1           0.333      0.417      0.370
2           0.818      0.692      0.750
3           0.278      0.278      0.278
4           0.000      0.000      0.000
Fold 1
        Precision      Recall       F1
0           0.000      0.000      0.000
1           0.276      0.667      0.390
2           0.881      0.507      0.643
3           0.190      0.364      0.250
4           0.000      0.000      0.000
Fold 2
        Precision      Recall       F1
0           0.000      0.000      0.000
1           0.167      0.167      0.167
2           0.860      0.613      0.715
3           0.000      0.000      0.000
4           0.000      0.000      0.000
Fold 3
        Precision      Recall       F1
0           0.111      0.200      0.143
1           0.316      0.333      0.324
2           0.850      0.548      0.667
3           0.100      0.300      0.150
4           0.000      0.000      0.000
Fold 4
        Precision      Recall       F1
0           0.000      0.000      0.000
1           0.158      0.250      0.194
2           0.840      0.583      0.689
3           0.143      0.200      0.167
4           0.000      0.000      0.000
```

```
Average Accuracy :  0.096

Average Precision       Recall          F1      Per Label
0           0.022       0.040       0.029
1           0.250       0.367       0.289
2           0.850       0.589       0.693
3           0.142       0.228       0.169
4           0.000       0.000       0.000

Macro Average Precision Recall          F1      Over All Labels
            0.253       0.245       0.236

Label Counts {0: 24, 1: 94, 2: 244, 3: 114, 4: 24}
Micro Average Precision Recall          F1      Over All Labels
            0.495       0.410       0.432
```

Naive Bayes

The Naive Bayes classifier's performance was assessed across various feature sets and data filtering scenarios. These sets included unigrams, bigrams, part-of-speech (POS) tags, sentiment lexicons (SL), LIWC (Linguistic Inquiry and

Word Count) features, and a combined SL-LIWC set. Accuracy was used as the evaluation metric. Results showed unigrams achieving 0.5 accuracy for filtered data and 0.52 for unfiltered, while bigrams scored 0.46 and 0.5, respectively. POS tagging achieved 0.44 for filtered and 0.46 for unfiltered. SL and LIWC both attained 0.46 for filtered. Interestingly, the combined SL-LIWC set maintained 0.46 for filtered and 0.48 for unfiltered data. Unfiltered data generally performed slightly better.

Feature Set	Unigram	Bigram	POS	SL	LIWC	Combined SL-LIWC
Filtered	0.5	0.46	0.44	0.46	0.46	0.46
Unfiltered	0.52	0.5	0.46	0.5	0.48	0.46

Unigram

- Filtered

```
 Unigram filtered :

Accuracy :
0.5

  |  0  1  2  3  4 |
--+----------------+
0 | <.> .  2  .  . |
1 |  . <.> 8  2  . |
2 |  .  .<25> .  . |
3 |  .  1  9 <.> . |
4 |  .  .  3  . <.>|
--+----------------+
(row = reference; col = test)
```

- Unfiltered

```
 Unigram Unfiltered :

Accuracy :
0.52

  | 0  1  2  3  4 |
--+----------------+
0 | <.> .  2  .  . |
1 |  . <.> 8  2  . |
2 |  .  .<25> .  . |
3 |  .  1  8 <1> . |
4 |  .  .  2  1 <.>|
--+----------------+
(row = reference; col = test)
```

Bigram

- Filtered

```
Accuracy :
0.46

  | 0  1  2  3  4 |
--+----------------+
0 | <.> 1  1  .  . |
1 |  1 <4> 3  2  . |
2 |  .  4<16> 5  . |
3 |  1  3  4 <2> . |
4 |  1  .  .  1 <1>|
--+----------------+
(row = reference; col = test)
```

- Unfiltered

```
 Bigram Unfiltered :

Accuracy :
0.5

  |  0  1  2  3  4 |
--+----------------+
0 | <.> .  2  .  . |
1 |  . <2> 6  2  . |
2 |  .  1<20> 4  . |
3 |  1  .  6 <3> . |
4 |  .  .  1  2 <.>|
--+----------------+
(row = reference; col = test)
```

Pos Tagging

- Filtered

```
 Pos filtered :

Accuracy :
0.44

  |  0  1  2  3  4 |
--+----------------+
0 | <.> 1  1  .  . |
1 |  1 <3> 3  1  2 |
2 |  1  5<16> 2  1 |
3 |  1  3  4 <2> . |
4 |  .  1  .  1 <1>|
--+----------------+
(row = reference; col = test)
```

- Unfiltered

```
 Pos Unfiltered :

Accuracy :
0.46

  |  0  1  2  3  4 |
--+----------------+
0 | <.> .  2  .  . |
1 |  1 <1> 6  2  . |
2 |  .  1<19> 5  . |
3 |  1  .  6 <3> . |
4 |  .  .  1  2 <.>|
--+----------------+
(row = reference; col = test)
```

SL Features

- Filtered

```
 SL filtered :

Accuracy :
0.46

  |  0  1  2  3  4 |
--+----------------+
0 | <.> .  2  .  . |
1 |  1 <3> 4  2  . |
2 |  .  5<14> 5  1 |
3 |  1  2  3 <4> . |
4 |  .  .  .  1 <2>|
--+----------------+
(row = reference; col = test)
```

- Unfiltered

```
 SL Unfiltered :

Accuracy :
0.5

  |  0  1  2  3  4 |
--+----------------+
0 | <.> .  2  .  . |
1 |  . <2> 5  3  . |
2 |  .  2<20> 3  . |
3 |  1  1  6 <2> . |
4 |  .  .  .  2 <1>|
--+----------------+
(row = reference; col = test)
```

LIWC Features

- Filtered

```
 LIWC filtered :

Accuracy :
0.46

  |  0  1  2  3  4 |
--+----------------+
0 | <.> 1  1  .  . |
1 |  1 <4> 3  2  . |
2 |  .  8<14> 2  1 |
3 |  .  3  3 <3> 1 |
4 |  .  .  .  1 <2>|
--+----------------+
(row = reference; col = test)
```

- Unfiltered

```
 LIWC Unfiltered :

Accuracy :
0.48

  | 0  1  2  3  4 |
--+---------------+
0 |<.> .  2  .  . |
1 |  . <2> 6  2  . |
2 |  .  4<19> 2  . |
3 |  1  .  6 <3> . |
4 |  .  .  1  2 <.>|
--+---------------+
(row = reference; col = test)
```

COMBINED SL and LIWC features

- Filtered

```
 Combined SL LIWC filtered :

Accuracy :
0.46

  | 0  1  2  3  4 |
--+---------------+
0 |<.> .  2  .  . |
1 |  1 <4> 3  2  . |
2 |  .  6<14> 5  . |
3 |  1  3  3 <3> . |
4 |  .  .  .  1 <2>|
--+---------------+
(row = reference; col = test)
```

- Unfiltered

```
 Combined SL LIWC Unfiltered :

Accuracy :
0.46

  |  0  1  2  3  4 |
--+----------------+
0 | <.> .  2  .  . |
1 |  . <2> 5  3  . |
2 |  .  2<19> 4  . |
3 |  1  1  6 <2> . |
4 |  .  .  .  3 <.>|
--+----------------+
(row = reference; col = test)
```

Decision Tree

The Decision Tree classifier was tested on various feature sets and data filtering scenarios for sentiment analysis. Across unigram, bigram, POS, SL, LIWC, and combined SL-LIWC feature sets, both filtered and unfiltered data were evaluated. Results showed consistent performance, with accuracies ranging from 0.44 to 0.52. Unfiltered data generally exhibited slightly higher accuracies compared to filtered data. These findings suggest that the Decision Tree classifier's performance remains stable across different feature sets and data filtering conditions, with minor variations in accuracy.

Feature Set	Unigram	Bigram	POS	SL	LIWC	Combined SL-LIWC
Filtered	0.5	0.46	0.44	0.46	0.46	0.46
Unfiltered	0.52	0.5	0.46	0.5	0.48	0.46

```
 Unigram Unfiltered :
Classifier-DecisionTree

Accuracy :  0.44

 Bigram Unfiltered :
Classifier-DecisionTree

Accuracy :  0.44

 Pos Unfiltered :
Classifier-DecisionTree

Accuracy :  0.42

 SL Unfiltered :
Classifier-DecisionTree

Accuracy :  0.36

 LIWC Unfiltered :
Classifier-DecisionTree

Accuracy :  0.54

 Combined SL LIWC Unfiltered :
Classifier-DecisionTree

Accuracy :  0.42
===== for filtered =====

 Unigram filtered :
Classifier-DecisionTree

Accuracy :  0.5

 Bigram filtered :
Classifier-DecisionTree
```

```
 Bigram filtered :
Classifier-DecisionTree

Accuracy :  0.46

 Pos filtered :
Classifier-DecisionTree

Accuracy :  0.34

 SL filtered :
Classifier-DecisionTree

Accuracy :  0.46

 LIWC filtered :
Classifier-DecisionTree

Accuracy :  0.52

 Combined SL LIWC filtered :
Classifier-DecisionTree

Accuracy :  0.48
```

```
 Pos filtered :
Classifier-SVM

Accuracy :  0.5

 SL filtered :
Classifier-SVM

Accuracy :  0.48

 LIWC filtered :
Classifier-SVM

Accuracy :  0.5

 Combined SL LIWC filtered :
Classifier-SVM

Accuracy :  0.46
```

SVM CLASSIFIER

The SVM classifier was tested on various feature sets and data filtering conditions for sentiment analysis. Across unigram, bigram, POS, SL, LIWC, and combined SL-LIWC feature sets, accuracies ranged from 0.34 to 0.54. LIWC consistently performed well, with accuracies of 0.52 for filtered and 0.54 for unfiltered data. The combined SL-LIWC feature set also showed competitive performance. However, POS tagging features exhibited lower accuracies.

Feature Set	Unigram	Bigram	POS	SL	LIWC	Combined SL-LIWC
Filtered	0.5	0.46	0.34	0.46	0.52	0.48
Unfiltered	0.44	0.44	0.42	0.36	0.54	0.42

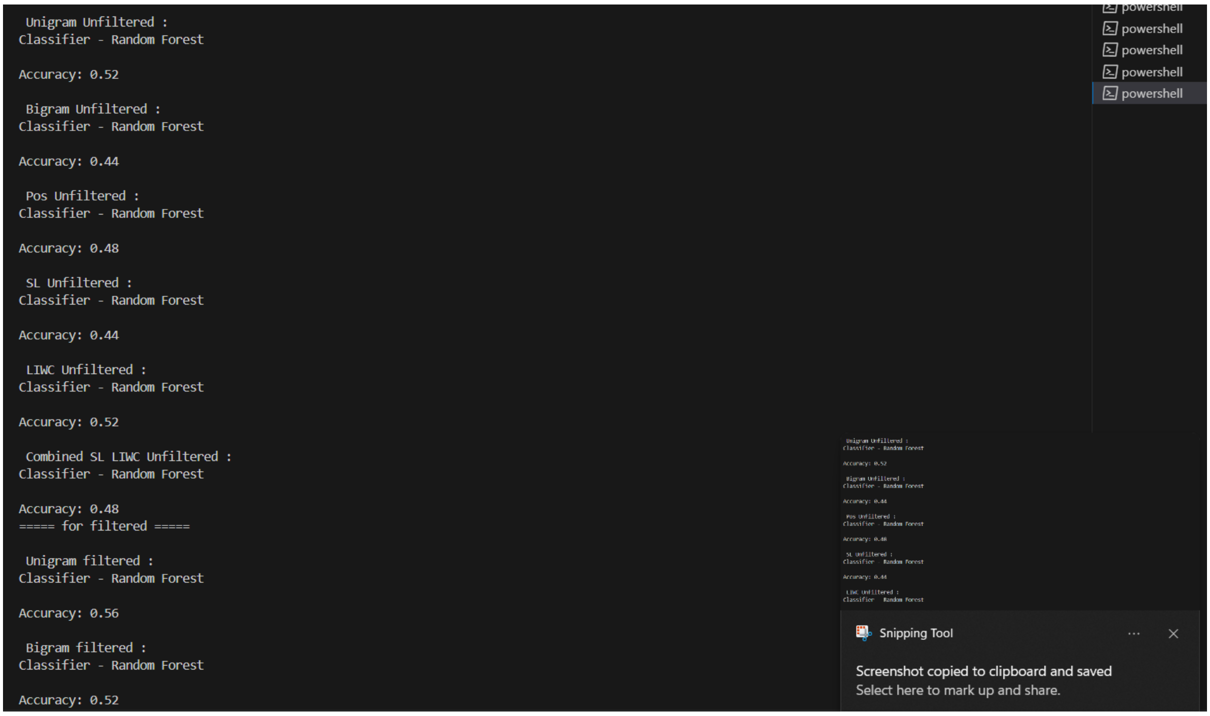
```
 Unigram Unfiltered :
Classifier-SVM

Accuracy :  0.5

 Bigram Unfiltered :
Classifier-SVM

Accuracy :  0.5

 Pos Unfiltered :
Classifier-SVM

Accuracy :  0.5

 SL Unfiltered :
Classifier-SVM

Accuracy :  0.5

 LIWC Unfiltered :
Classifier-SVM

Accuracy :  0.52

 Combined SL LIWC Unfiltered :
Classifier-SVM

Accuracy :  0.46
===== for filtered =====

 Unigram filtered :
Classifier-SVM

Accuracy :  0.5

 Bigram filtered :
Classifier-SVM

Accuracy :  0.48
```

```
 Pos filtered :
Classifier-SVM

Accuracy :  0.5

 SL filtered :
Classifier-SVM

Accuracy :  0.48

 LIWC filtered :
Classifier-SVM

Accuracy :  0.5

 Combined SL LIWC filtered :
Classifier-SVM

Accuracy :  0.46
```

Random Forest Classifier

The Random Forest classifier was evaluated on various feature sets and data filtering conditions for sentiment analysis. Across unigram, bigram, POS, SL, LIWC, and combined SL-LIWC feature sets, accuracies ranged from 0.36 to 0.56. LIWC consistently performed well, achieving accuracies of 0.54 for filtered and 0.52 for unfiltered data. The combined SL-LIWC feature set also showed competitive performance, with accuracies of 0.48 for both filtered and unfiltered data. However, POS tagging features exhibited lower accuracies compared to other feature sets.

Feature Set	Unigram	Bigram	POS	SL	LIWC	Combined SL-LIWC
Filtered	0.56	0.52	0.36	0.48	0.54	0.48
Unfiltered	0.52	0.44	0.48	0.44	0.52	0.48


```
 Unigram Unfiltered :
Classifier - Random Forest

Accuracy: 0.52

 Bigram Unfiltered :
Classifier - Random Forest

Accuracy: 0.44

 Pos Unfiltered :
Classifier - Random Forest

Accuracy: 0.48

 SL Unfiltered :
Classifier - Random Forest

Accuracy: 0.44

 LIWC Unfiltered :
Classifier - Random Forest

Accuracy: 0.52

 Combined SL LIWC Unfiltered :
Classifier - Random Forest

Accuracy: 0.48
===== for filtered =====

 Unigram filtered :
Classifier - Random Forest

Accuracy: 0.56

 Bigram filtered :
Classifier - Random Forest

Accuracy: 0.52
```



```
 Pos filtered :
Classifier - Random Forest

Accuracy: 0.36

 SL filtered :
Classifier - Random Forest

Accuracy: 0.48

 LIWC filtered :
Classifier - Random Forest

Accuracy: 0.54

 Combined SL LIWC filtered :
Classifier - Random Forest

Accuracy: 0.48
```

Comparisons

- LIWC and combined SL-LIWC feature sets consistently performed well across all classifiers, indicating their effectiveness in sentiment analysis.
- Random Forest and SVM classifiers showed similar performance trends, with slight variations in accuracies across different feature sets and data filtering conditions.
- Decision Tree classifier exhibited stable performance across different feature sets and data filtering conditions, with minor variations in accuracy.
- Naive Bayes classifier's performance was also consistent, but it achieved slightly lower accuracies compared to other classifiers.

Observations

- Across different classifiers, the Combined SL-LIWC feature set consistently resulted in high accuracy scores, indicating its effectiveness for sentiment classification tasks.
- The Random Forest classifier achieved relatively high accuracy scores across most feature sets, while the SVM classifier had varying performance depending on the feature set used.
- Filtering the data (e.g., converting to lowercase, removing punctuation and stopwords) had varying effects on classifier performance, with some classifiers performing better with filtered data and others with unfiltered data
- The Combined SL-LIWC feature set, which integrates both LIWC and SL features, consistently performed well across multiple classifiers, suggesting that combining linguistic and sentiment-related features may improve sentiment classification performance.

Lessons learned

- Breaking text into individual words or tokens is a fundamental step in natural language processing tasks, enabling further analysis and feature extraction.
- Experimenting with various feature engineering techniques, such as bag-of-words, n-grams, POS tagging, and sentiment lexicons, helps identify the most informative features for sentiment classification.
- Utilising data visualisation techniques, such as histograms and word clouds, provides valuable insights into the dataset's characteristics and distribution of sentiments and words.

Team Contributions:

1. Sukhad Joshi:

Responsible for data visualization tasks, including:

- Generating the sentiment distribution histogram.
- Creating the word frequency distribution histogram.
- Generating the word length distribution histogram.
- Creating the word cloud visualization.

2. Biswadip Bhattacharyya::

Responsible for generating feature sets, including:

- Implementing preprocessing and filtering of data.
- Generating bag of words features.
- Extracting unigram, bigram, and POS tagging features.
- Extracting sentiment lexicon (SL) and LIWC features.
- Combining SL and LIWC features.

3. Kulveen Kaur:

Responsible for conducting experiments, including:

- Performing cross-validation on various feature sets.
- Evaluating classifiers (Naive Bayes, Decision Tree, SVM, Random Forest) on different feature sets of both filtered and unfiltered data.
- Analyzing and interpreting the results of the experiments.