Project Report on

A Novel Approach to Animosity Detection

Submitted in partial fulfillment of the requirements of

the degree of Bachelor in Engineering

by

Name of student	Class	Roll No.
Omkar Dalvi	BE-3	5
Rutvik Deshpande	BE-3	7
Aayush Joshi	BE-3	15
Harsh Gala	BE-3	12

Under the guidance of

Mr. Atul Kachare

DEPARTMENT OF COMPUTER ENGINEERING

SHAH AND ANCHOR KUTCHHI ENGINEERING COLLEGE

CHEMBUR, MUMBAI – 400088.

2020 – 2021

Problem Statement:

Our task is to generate a lexicon of sentiment expressions using semantic and subjectivity features with an orientation to hate speech and then use these features to create a classifier for hate speech detection

Technologies Used:

Python, SentiWordNet (Indian Languages: Hindi), SYNSET, Stanza

Theory:

1. Python:
   Python is a high-level,general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.

2. SentiWordNet:
   SentiWordNet is a lexical resource for opinion mining. SentiWordNet assigns to each synset of WordNet three sentiment scores: positivity, negativity, objectivity.

3. Stanza:
   Stanza is a collection of accurate and efficient tools for the linguistic analysis of many human languages.

4. Synset:
   In order to search up terms in WordNet, Synset is a specific type of simple interface that is available in NLTK. Synset instances are collections of words that communicate the same idea and are synonyms. Some of the words have only one Synset and some have several.

5. The corpus:
   People appear to have strong emotional attachments to their views and political and social topics are very divisive, which leads to a lot of rudeness and hate speech in threads (on, example, Twitter) that address political matters. Because of this, a lot of the tweets in our corpus are political or social in character and contain hate speech. If you visit the corpus (mentioned above), you will see that it has been divided up into sub-categories that specify the level of hatred in the tweets. This was accomplished using a comma-separated values (csv) file, where each row contains an index number, the tweet in question, and a categorization for its "hatefulness."
   The categories are:
   • Non-hostile
   • Fake
   • Defamation
   • Offensive
   • Hate

Naturally, not all of them are mutually disjoint, hence the corpus enlists various categories for some tweets instead of just one. For example, a tweet that's fake could also be defamatory as well as offensive. It is important to note here that all the tweets marked "non-hostile" are devoid of any malintent and thus do not fall into any of the other categories.

Literature Survey:

Paper Details	Methodology used	Comment
A Measurement Study of Hate Speech in Social Media.	Extraction of contents in keywords, they use TF-IDF score, which they use to rank the feature vectors. Different from these studies, we are concerned with content analysis of hate speech using sentiment analysis techniques.	The author uses a combination of text mining techniques and social network analysis to locate and understand how various hate groups in Facebook share their ideas and attract new users. Using text mining techniques, they extract the keywords that are frequently used within the groups and rely on social network structure to find their relations.
Filtering offensive language in online communities using grammatical relations.	Their approach is centered on what they term as modification and pattern integrity heuristic rules. However, their researches are concerned with offensive messages in general and not hate speech as in our case.	The authors have used grammatical relations among words to semantically remove offensive contents in a sentence. For each sentence, they begin by identifying offensive words, and then they extract semantic and syntactic relations among words in the sentence
Detecting Hate Speech on the World Wide Web.	Naïve Bayes classifier is used to distinguish between racist and non-racists tweets against black people. The approach focuses exclusively on unigram features and is therefore a text classification application.	The authors have presented a supervised approach that categorizes hate speech by identifying stereotypes used in the text. Some of the categories identified include anti-Semitic, anti-Muslim and anti-African.
A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts.	Then, using minimum cuts formulation, they integrate in sentence level contextual information with traditional bag-of-words features.	The author used a subjectivity detector to remove objective sentences from a given document.

Methodology:

Our approach consists of three major steps.

- Subjectivity analysis
- Building a hate speech lexicon
- Identifying theme-based nouns.

Subjectivity analysis

Subjectivity analysis helps us reduce the corpus since only subjective tweets will contain hate speech rather than, say, facts. It is therefore quite helpful to seek for impartial tweets and designate them as "not hate." We must first grade subjectivity and eliminate the objective statements in order to do this. SentiWordNet is used for this, and it allows us to give each word a positive and negative score. Then, we match each key in the newly developed SUBJCLUE lexicon to its presence in our dataset and determine the appropriate negative and positive scores. The overall score for that tweet or statement is then calculated using the scores. We determined that any tweets with a score above 1.0 or below -0.5 were arbitrary enough to be significant contributors to hate speech. We selected a range of values as the limiting values that make a tweet worthwhile studying further. Because we discovered that phrases identified negatively by word sentiment analysis were more likely to include hate speech, the lower limit is quantitatively lower than the higher one.

Building a lexicon for hate speech

Now that the subjective sentences have been separated, we can begin creating a vocabulary that contains phrases that will enable us to more precisely detect hate speech. Then first compile a list of all the terms from the SUBJCLUE vocabulary that are strongly negative and have a total score of less than -0.25, and we do the same for words that are negatively inclined but have a score more than -0.25. These lists assist us in identifying hate speech and negative polarity in phrases and tweets. After carefully reviewing the verbs in the dataset that we considered to have a hostile meaning or to be mostly used for spiteful remarks but were not included in the SUBJCLUE, we next employ an initial list of seed verbs, consisting of verbs manually picked. We then use a SYNSET to find all the possible synonyms of these verbs, and add all of those to a list of hateful verbs. After having this hate-verb list built, we check which of those hate-verbs occur in the hate corpus and if they do then we add them into a final hate-verbs lexicon.

Theme-based nouns

Now that we have lexicons for hateful words and verbs in general, we need to make sure that they are presented with nouns that are pertinent to the discussion - like curse words and theme-dependent nouns (like politically significant words) - to make sure that they are used in messages that spew hatred. To do this, we first divide the corpus into noun phrases and list the most common NPs that are included in tweets classified as nasty. A text file containing a list of a few dozen such recurring nouns is added to for subsequent use when testing for hate speech.

Code:

Open in Colab

Code:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```

```
SUBJCLUE = []

with open('/content/drive/MyDrive/Projects/NLP Project/project/SUBJCLUE.txt') as f:
    for line in f:
      x = line.split()
      x[4] = x[4].split(',')
      SUBJCLUE.append(x)


for key in SUBJCLUE[:5]:
  print(key[4])
```

```
['अनौपचारिक']
['मृत']
['परवर्ती']
['अच्छा', 'बढ़िया']
['सौभाग्यशाली', 'खुशकिस्मत', 'खुशनसीब', 'तक़दीर_वाला', 'नसीब_वाला', 'भाग्यवान', 'भाग्यशाली', 'ख़ुश
```

Reading the data

```
import csv

filename = "/content/drive/MyDrive/Projects/NLP Project/project/Dataset/valid.csv"

fields = []
rows = []

with open(filename, 'r') as csvfile:

    csvreader = csv.reader(csvfile)


    fields = next(csvreader)
    for row in csvreader:
        rows.append(row)


    print("Total no. of rows: %d"%(csvreader.line_num))


print('Field names are:' + ', '.join(field for field in fields))


tot = 0
for row in rows:
    row.append(tot)
for row in rows[:5]:
  print(row)
```

```
Total no. of rows: 2259
Field names are:Unique ID, Post, Labels Set
['1', 'दृढ़ इच्छा शक्ति से परिपूर्ण प्रणबदा के लिए देशहित सर्वोच्च रहा।\n\nउनका निधन हम सब के लिए अपूरण
['2', 'भारतीय जनता पार्टी rss वाले इतने गिरे हुए हैं जहां मैं रहती हूं वहां मेरी जासूसी  करा रहें है उसकी जार
['3', 'कोरोना से निपटने की तैयारी / दिल्ली में 10 हजार बेड वाला दुनिया का सबसे बड़ा कोविड केयर सेंटर शु
['4', 'गवर्नर कॉन्फ्रेंस में PM मोदी बोले- शिक्षा नीति में सरकार का दखल कम होना चाहिए\nhttps://t.co/Zv
['5', 'यूपी: गाजीपुर में Toilet घोटाला, प्रधान व सचिव ने किया लाखों का गबन, मुर्दों के नाम पर बनवा डाले
```

Checking score

Finding positive, negative and total scores for each sentence

```
count = 0
for key in SUBJCLUE:
  subjlist = key[4]
  for row in rows:
    if any([subjword in row[1] for subjword in subjlist]):
      count += 1
      pos = float(key[2])
      neg = float(key[3])
      tot = pos - neg
      row[3] += tot

print(count)
```

```
10530
```

Hate Lexicon Growing

```
!pip install stanza
```

```
    Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
    Requirement already satisfied: stanza in /usr/local/lib/python3.7/dist-packages (1.4.2)
    Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from stan
    Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from sta
    Requirement already satisfied: torch>=1.3.0 in /usr/local/lib/python3.7/dist-packages (
    Requirement already satisfied: emoji in /usr/local/lib/python3.7/dist-packages (from sta
    Requirement already satisfied: protobuf in /usr/local/lib/python3.7/dist-packages (from
    Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from
    Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from star
    Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packag
    Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packag
    Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lil
    Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packa
    Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (
```

```
SYNSET = []
with open('/content/drive/MyDrive/Projects/NLP Project/project/Synset.txt', encoding= 'unicod
    for line in f:
        x = line.split()
        x[3] = x[3].split(':')
        SYNSET.append(x)

import stanza
stanza.download('hi', processors='tokenize,pos,lemma')

import csv
dataset = ""

for row in rows:
    dataset+=row[1]

verbs_content = []
nlp = stanza.Pipeline('hi',processors='tokenize,pos,lemma')
doc = nlp(dataset)
for sentence in doc.sentences:
     for word in sentence.words:
         if word.upos == 'VERB':
             verbs_content.append(word.text)

strongly_negative_words = []
weakly_negative_words = []
for line in SUBJCLUE:
    totalscore = float(line[2]) - float(line[3])
    if(totalscore < -0.25):
      for word in line[4]:
        strongly_negative_words.append(word)
    elif totalscore < 0:
      for word in line[4]:
        weakly_negative_words.append(word)

def Getsynset(word):
    syn = []
    flag=0
    syn.append(word)
    for line in SYNSET:
        if(line[1]=="03"):
            for verb in line[3]:
                if(word == verb):
                    flag = 1
                    break
            if(flag):
                syn = line[3]
                break
    return syn

s = {}
hlex = []

slist = ["लड़ना" , "मारना" , "लूटना" , "पीटना" , "कूटना" , "भेदभाव" ,"फोड़ना", "तोड़ना", "उखाड़ना" ]
for word in slist:
  hlex.append(word)
for word in slist:
    s = Getsynset(word)
    for verb1 in s:
        if verb1 in verbs_content:
            hlex.append(verb1)
```

```
Downloading                                                     193k/?
https://raw.githubusercontent.com/stanfordnlp/stanza-           [00:00<00:00,
resources/main/resources_1.4.1.json:                            4.50MB/s]
INFO:stanza:Downloading these customized packages for language: hi (Hindi)...
=============================
| Processor       | Package |
-----------------------------
| tokenize        | hdtb    |
| pos             | hdtb    |
| lemma           | hdtb    |
| forward_charlm  | oscar   |
| pretrain        | hdtb    |
| backward_charlm | oscar   |
```

```
themed_nouns = open('/content/drive/MyDrive/Projects/NLP Project/project/themenouns.txt','r')
themenouns = []
for line in themed_nouns:
    themenouns.append(line.rstrip('\n'))
print(themenouns)
```

```
['बीजेपी ', 'मोदी ', 'माओवादियों ', 'इस्लाम ', 'धमकी ', 'सुरक्षा ', 'धर्म ', 'साले ', 'कुत्ते ', 'कुि
Downloading                                                     193k/?
```

Hate speech Detection Algorithm

```
INFO:stanza:Loading these models for language: hi (Hindi):
```

```
print("strongly_negative_words",strongly_negative_words)
print("weakly_negative_words",weakly_negative_words)
print("hate lex",hlex)
print("theme nouns",themenouns)
```

```
strongly_negative_words ['मृत', 'दुर्भाग्यशाली', 'अभागा', 'बदनसीब', 'भाग्यहीन', 'मनहूस', 'बदकि
weakly_negative_words ['अच्छा', 'बढ़िया', 'कठिनाई_से', 'जैसे_तैसे', 'मुश्किल_से', 'कठिनतः', 'सड़
hate lex ['लड़ना', 'मारना', 'लूटना', 'पीटना', 'कूटना', 'भेदभाव', 'फोड़ना', 'तोड़ना', 'उखाड़ना',
theme nouns ['बीजेपी ', 'मोदी ', 'माओवादियों ', 'इस्लाम ', 'धमकी ', 'सुरक्षा ', 'धर्म ', 'साले ',
```

Calculating Scores without Subjective Analysis

Only Semantic feature set

```
for row in rows:
    strongcount = 0
    hlexcount = 0
    weakcount = 0
    themecount = 0
    if any([word in row[1] for word in strongly_negative_words]):
      strongcount += 1
    if any([word in row[1] for word in weakly_negative_words]):
      weakcount += 1

    if strongcount >= 2:
        row.append("strongly hateful")
    elif strongcount == 1:
      if hlexcount >= 1 or themecount >= 1:
        row.append("strongly hateful")
      else:
        row.append("weakly hateful")
    elif strongcount == 0:
      if themecount >= 1 and hlexcount >= 1:
        row.append("strongly hateful")
      elif themecount >=1 and weakcount >= 1:
        row.append("weakly hateful")
      elif hlexcount == 1:
        row.append("weakly hateful")
      else:
        row.append("No Hate")



total_rows = [row for row in rows]


no_hate_rows = [row for row in rows if row[4] == "No Hate"]

correct_no_hate_rows = [row for row in no_hate_rows if row[4] == "No Hate" and row[2] == "non

weak_hate_rows = [row for row in rows if row[4] == "weakly hateful"]

correct_weak_hate_rows = [row for row in weak_hate_rows if row[4] == "weakly hateful" and (ro

strong_hate_rows = [row for row in rows if row[4] == "strongly hateful"]

correct_strong_hate_rows = [row for row in strong_hate_rows if row[4] == "strongly hateful" a

false_neg_no_hate = [row for row in no_hate_rows if row[2] == "non-hostile" and row[4] != "No

false_neg_weak_hate = [row for row in weak_hate_rows if row[2] == "fake" or row[2] == "defama

false_neg_strong_hate = [row for row in strong_hate_rows if row[2] != "non-hostile" and row[2

# calculating precision
precision = (len(correct_no_hate_rows)+len(correct_strong_hate_rows)+len(correct_weak_hate_ro
# calculating recall
recall = (len(correct_no_hate_rows)+len(correct_strong_hate_rows)+len(correct_weak_hate_rows)
# calculating F1 score
f1 = 2*precision*recall/(precision+recall)

print("Total no. of rows: {}".format(len(total_rows)))
print("No Hate: {}".format(len(no_hate_rows)))
print("Actual no hate: {}".format(len(correct_no_hate_rows)))
print("Weak Hate: {}".format(len(weak_hate_rows)))
print("Actual weak hate: {}".format(len(correct_weak_hate_rows)))
print("Strong Hate: {}".format(len(strong_hate_rows)))
print("Actual strong hate: {}".format(len(correct_strong_hate_rows)))

print("Precision: {}".format(precision))

print("Recall: {}".format(recall))

print("F-score: {}".format(f1))
```

```
Total no. of rows: 811
No Hate: 357
Actual no hate: 207
Weak Hate: 454
Actual weak hate: 112
Strong Hate: 0
Actual strong hate: 0
Precision: 0.3933415536374846
Recall: 0.7799511002444988
F-score: 0.5229508196721312
```

Semantic + Hate Lexicon

```
for row in rows:
    strongcount = 0
    hlexcount = 0
    weakcount = 0
    themecount = 0
    if any([word in row[1] for word in strongly_negative_words]):
      strongcount += 1
    if any([word in row[1] for word in hlex]):
      hlexcount += 1
    if any([word in row[1] for word in weakly_negative_words]):
      weakcount += 1

    if strongcount >= 2:
        row[4] = "strongly hateful"
    elif strongcount == 1:
      if hlexcount >= 1 or themecount >= 1:
        row[4] = "strongly hateful"
      else:
        row[4] = "weakly hateful"
    elif strongcount == 0:
      if themecount >= 1 and hlexcount >= 1:
        row[4] = "strongly hateful"
      elif themecount >=1 and weakcount >= 1:
        row[4] = "weakly hateful"
      elif hlexcount == 1:
        row[4] = "weakly hateful"
      else:
        row[4] = "No Hate"



total_rows = [row for row in rows]


no_hate_rows = [row for row in rows if row[4] == "No Hate"]

correct_no_hate_rows = [row for row in no_hate_rows if row[4] == "No Hate" and row[2] == "non

weak_hate_rows = [row for row in rows if row[4] == "weakly hateful"]

correct_weak_hate_rows = [row for row in weak_hate_rows if row[4] == "weakly hateful" and (ro

strong_hate_rows = [row for row in rows if row[4] == "strongly hateful"]

correct_strong_hate_rows = [row for row in strong_hate_rows if row[4] == "strongly hateful" a

false_neg_no_hate = [row for row in no_hate_rows if row[2] == "non-hostile" and row[4] != "No

false_neg_weak_hate = [row for row in weak_hate_rows if row[2] == "fake" or row[2] == "defama

false_neg_strong_hate = [row for row in strong_hate_rows if row[2] != "non-hostile" and row[2

# calculating precision
precision = (len(correct_no_hate_rows)+len(correct_strong_hate_rows)+len(correct_weak_hate_ro
# calculating recall
recall = (len(correct_no_hate_rows)+len(correct_strong_hate_rows)+len(correct_weak_hate_rows)
# calculating F1 score
f1 = 2*precision*recall/(precision+recall)

print("Total no. of rows: {}".format(len(total_rows)))
print("No Hate: {}".format(len(no_hate_rows)))
print("Actual no hate: {}".format(len(correct_no_hate_rows)))
print("Weak Hate: {}".format(len(weak_hate_rows)))
print("Actual weak hate: {}".format(len(correct_weak_hate_rows)))
print("Strong Hate: {}".format(len(strong_hate_rows)))
print("Actual strong hate: {}".format(len(correct_strong_hate_rows)))

print("Precision: {}".format(precision))

print("Recall: {}".format(recall))
print("F-score: {}".format(f1))
```

```
Total no. of rows: 811
No Hate: 355
Actual no hate: 206
Weak Hate: 455
Actual weak hate: 112
Strong Hate: 1
Actual strong hate: 1
Precision: 0.3933415536374846
Recall: 0.7799511002444988
F-score: 0.5229508196721312
```

Semantic + Hate Lexicon + Thematic Nouns

```
for row in rows:
    strongcount = 0
    hlexcount = 0
    weakcount = 0
    themecount = 0
    if any([word in row[1] for word in strongly_negative_words]):
      strongcount += 1
    if any([word in row[1] for word in hlex]):
      hlexcount += 1
    if any([word in row[1] for word in weakly_negative_words]):
      weakcount += 1
    if any([word in row[1] for word in themenouns]):
      themecount += 1
    if strongcount >= 2:
        row[4] = "strongly hateful"
    elif strongcount == 1:
      if hlexcount >= 1 or themecount >= 1:
        row[4] = "strongly hateful"
      else:
        row[4] = "weakly hateful"
    elif strongcount == 0:
      if themecount >= 1 and hlexcount >= 1:
        row[4] = "strongly hateful"
      elif themecount >=1 and weakcount >= 1:
        row[4] = "weakly hateful"
      elif hlexcount == 1:
        row[4] = "weakly hateful"
      else:
        row[4] = "No Hate"



total_rows = [row for row in rows]
```

```
no_hate_rows = [row for row in rows if row[4] == "No Hate"]

correct_no_hate_rows = [row for row in no_hate_rows if row[4] == "No Hate" and row[2] == "non

weak_hate_rows = [row for row in rows if row[4] == "weakly hateful"]

correct_weak_hate_rows = [row for row in weak_hate_rows if row[4] == "weakly hateful" and (ro

strong_hate_rows = [row for row in rows if row[4] == "strongly hateful"]

correct_strong_hate_rows = [row for row in strong_hate_rows if row[4] == "strongly hateful" a

false_neg_no_hate = [row for row in no_hate_rows if row[2] == "non-hostile" and row[4] != "No

false_neg_weak_hate = [row for row in weak_hate_rows if row[2] == "fake" or row[2] == "defama

false_neg_strong_hate = [row for row in strong_hate_rows if row[2] != "non-hostile" and row[2

# calculating precision
precision = (len(correct_no_hate_rows)+len(correct_strong_hate_rows)+len(correct_weak_hate_ro
# calculating recall
recall = (len(correct_no_hate_rows)+len(correct_strong_hate_rows)+len(correct_weak_hate_rows)
# calculating F1 score
f1 = 2*precision*recall/(precision+recall)

print("Total no. of rows: {}".format(len(total_rows)))
print("No Hate: {}".format(len(no_hate_rows)))
print("Actual no hate: {}".format(len(correct_no_hate_rows)))
print("Weak Hate: {}".format(len(weak_hate_rows)))
print("Actual weak hate: {}".format(len(correct_weak_hate_rows)))
print("Strong Hate: {}".format(len(strong_hate_rows)))
print("Actual strong hate: {}".format(len(correct_strong_hate_rows)))

print("Precision: {}".format(precision))

print("Recall: {}".format(recall))

print("F-score: {}".format(f1))
```

```
Total no. of rows: 811
No Hate: 293
Actual no hate: 185
Weak Hate: 344
Actual weak hate: 85
Strong Hate: 174
Actual strong hate: 75
Precision: 0.4254007398273736
Recall: 0.843520782396088
F-score: 0.5655737704918032
```

Calculating Scores with Subjective Analysis

```
counter = 0
subj_rows = []
for row in rows:
  if row[3] <= -0.5 or row[3] >= 1:
    subj_rows.append(row)
    counter += 1

print("Number of Subjective Sentences: ")
print(counter)
print(rows[0])
```

```
Number of Subjective Sentences:
355
['1', 'दृढ़ इच्छा शक्ति से परिपूर्ण प्रणबदा के लिए देशहित सर्वोच्च रहा।\n\nउनका निधन हम सब के लिए अपूरण
```

Semantic feature set

```
for row in rows:
  if row[3] <= -0.5 or row[3] >= 1:
    strongcount = 0
    hlexcount = 0
    weakcount = 0
    themecount = 0
    if any([word in row[1] for word in strongly_negative_words]):
      strongcount += 1

    if any([word in row[1] for word in weakly_negative_words]):
      weakcount += 1


    if strongcount >= 2:
        row.append("strongly hateful")
    elif strongcount == 1:
      if hlexcount >= 1 or themecount >= 1:
        row.append("strongly hateful")
      else:
        row.append("weakly hateful")
    elif strongcount == 0:
      if themecount >= 1 and hlexcount >= 1:
        row.append("strongly hateful")
      elif themecount >=1 and weakcount >= 1:
        row.append("weakly hateful")
      elif hlexcount == 1:
        row.append("weakly hateful")
      else:
        row.append("No Hate")
  else:
     row.append("No Hate")


total_rows = [row for row in rows]

no_hate_rows = [row for row in rows if row[5] == "No Hate"]
correct_no_hate_rows = [row for row in no_hate_rows if row[5] == "No Hate" and row[2] == "non
weak_hate_rows = [row for row in rows if row[5] == "weakly hateful"]
correct_weak_hate_rows = [row for row in weak_hate_rows if row[5] == "weakly hateful" and (ro
strong_hate_rows = [row for row in rows if row[5] == "strongly hateful"]
correct_strong_hate_rows = [row for row in strong_hate_rows if row[5] == "strongly hateful" a

false_neg_no_hate = [row for row in no_hate_rows if row[2] == "non-hostile" and row[5] != "No
false_neg_weak_hate = [row for row in weak_hate_rows if row[2] == "fake" or row[2] == "defama
false_neg_strong_hate = [row for row in strong_hate_rows if row[2] != "non-hostile" and row[2

precision = (len(correct_no_hate_rows)+len(correct_strong_hate_rows)+len(correct_weak_hate_ro
recall = (len(correct_no_hate_rows)+len(correct_strong_hate_rows)+len(correct_weak_hate_rows)
f1 = 2*precision*recall/(precision+recall)

print("Total no. of rows: {}".format(len(total_rows)))
print("No Hate: {}".format(len(no_hate_rows)))
print("Actual no hate: {}".format(len(correct_no_hate_rows)))
print("Weak Hate: {}".format(len(weak_hate_rows)))
print("Actual weak hate: {}".format(len(correct_weak_hate_rows)))
print("Strong Hate: {}".format(len(strong_hate_rows)))
print("Actual strong hate: {}".format(len(correct_strong_hate_rows)))
print("Precision: {}".format(precision))
print("Recall: {}".format(recall))
print("F-score: {}".format(f1))
```

```
Total no. of rows: 811
No Hate: 479
Actual no hate: 282
Weak Hate: 332
Actual weak hate: 86
Strong Hate: 0
Actual strong hate: 0
Precision: 0.45376078914919854
Recall: 0.8498845265588915
F-score: 0.5916398713826366
```

Semantic + Hate Lexicon

```
for row in rows:
  if row[3] <= -0.5 or row[3] >= 1:
    strongcount = 0
    hlexcount = 0
    weakcount = 0
    themecount = 0
    if any([word in row[1] for word in strongly_negative_words]):
      strongcount += 1
    if any([word in row[1] for word in hlex]):
      hlexcount += 1
    if any([word in row[1] for word in weakly_negative_words]):
      weakcount += 1

    if strongcount >= 2:
        row[5] = "strongly hateful"
    elif strongcount == 1:
      if hlexcount >= 1 or themecount >= 1:
        row[5] = "strongly hateful"
      else:
        row[5] = "weakly hateful"
    elif strongcount == 0:
      if themecount >= 1 and hlexcount >= 1:
        row[5] = "strongly hateful"
      elif themecount >=1 and weakcount >= 1:
        row[5] = "weakly hateful"
      elif hlexcount == 1:
        row[5] = "weakly hateful"
      else:
        row[5] = "No Hate"


total_rows = [row for row in rows]

no_hate_rows = [row for row in rows if row[5] == "No Hate"]
correct_no_hate_rows = [row for row in no_hate_rows if row[5] == "No Hate" and row[2] == "non
weak_hate_rows = [row for row in rows if row[5] == "weakly hateful"]
correct_weak_hate_rows = [row for row in weak_hate_rows if row[5] == "weakly hateful" and (ro
strong_hate_rows = [row for row in rows if row[5] == "strongly hateful"]
correct_strong_hate_rows = [row for row in strong_hate_rows if row[5] == "strongly hateful" a

false_neg_no_hate = [row for row in no_hate_rows if row[2] == "non-hostile" and row[5] != "No
false_neg_weak_hate = [row for row in weak_hate_rows if row[2] == "fake" or row[2] == "defama
false_neg_strong_hate = [row for row in strong_hate_rows if row[2] != "non-hostile" and row[2

precision = (len(correct_no_hate_rows)+len(correct_strong_hate_rows)+len(correct_weak_hate_ro
recall = (len(correct_no_hate_rows)+len(correct_strong_hate_rows)+len(correct_weak_hate_rows)
f1 = 2*precision*recall/(precision+recall)

print("Total no. of rows: {}".format(len(total_rows)))
print("No Hate: {}".format(len(no_hate_rows)))
print("Actual no hate: {}".format(len(correct_no_hate_rows)))
print("Weak Hate: {}".format(len(weak_hate_rows)))
print("Actual weak hate: {}".format(len(correct_weak_hate_rows)))
print("Strong Hate: {}".format(len(strong_hate_rows)))
print("Actual strong hate: {}".format(len(correct_strong_hate_rows)))
print("Precision: {}".format(precision))
print("Recall: {}".format(recall))
print("F-score: {}".format(f1))
```

```
Total no. of rows: 811
No Hate: 479
Actual no hate: 282
Weak Hate: 331
Actual weak hate: 86
Strong Hate: 1
Actual strong hate: 1
Precision: 0.45499383477188654
Recall: 0.8502304147465438
F-score: 0.5927710843373494
```

Semantic + Hate Lexicon + Thematic Nouns

```
def testoutput(rows):
  for row in rows:
    if row[3] <= -0.5 or row[3] >= 1:
      strongcount = 0
      hlexcount = 0
      weakcount = 0
      themecount = 0
      if any([word in row[1] for word in strongly_negative_words]):
        strongcount += 1
      if any([word in row[1] for word in hlex]):
        hlexcount += 1
      if any([word in row[1] for word in weakly_negative_words]):
        weakcount += 1
      if any([word in row[1] for word in themenouns]):
        themecount += 1

      if strongcount >= 2:
          row[5] = "strongly hateful"
      elif strongcount == 1:
        if hlexcount >= 1 or themecount >= 1:
          row[5] = "strongly hateful"
        else:
          row[5] = "weakly hateful"
      elif strongcount == 0:
        if themecount >= 1 and hlexcount >= 1:
          row[5] = "strongly hateful"
        elif themecount >=1 and weakcount >= 1:
          row[5] = "weakly hateful"
        elif hlexcount == 1:
          row[5] = "weakly hateful"
        else:
          row[5] = "No Hate"
  total_rows = [row for row in rows]

  no_hate_rows = [row for row in rows if row[-1] == "No Hate"]
  correct_no_hate_rows = [row for row in no_hate_rows if row[-1] == "No Hate" and row[2] == "
  weak_hate_rows = [row for row in rows if row[-1] == "weakly hateful"]
  correct_weak_hate_rows = [row for row in weak_hate_rows if row[-1] == "weakly hateful" and
  strong_hate_rows = [row for row in rows if row[-1] == "strongly hateful"]
  correct_strong_hate_rows = [row for row in strong_hate_rows if row[-1] == "strongly hateful

  false_neg_no_hate = [row for row in no_hate_rows if row[2] == "non-hostile" and row[-1] !=
  false_neg_weak_hate = [row for row in weak_hate_rows if row[2] == "fake" or row[2] == "defa
  false_neg_strong_hate = [row for row in strong_hate_rows if row[2] != "non-hostile" and row

  precision = (len(correct_no_hate_rows)+len(correct_strong_hate_rows)+len(correct_weak_hate_
  recall = (len(correct_no_hate_rows)+len(correct_strong_hate_rows)+len(correct_weak_hate_row
  f1 = 2*precision*recall/(precision+recall)

  print("Total no. of rows: {}".format(len(total_rows)))
  print("No Hate: {}".format(len(no_hate_rows)))
  print("Actual no hate: {}".format(len(correct_no_hate_rows)))
  print("Weak Hate: {}".format(len(weak_hate_rows)))
  print("Actual weak hate: {}".format(len(correct_weak_hate_rows)))
  print("Strong Hate: {}".format(len(strong_hate_rows)))
  print("Actual strong hate: {}".format(len(correct_strong_hate_rows)))
  print("Precision: {}".format(precision))
  print("Recall: {}".format(recall))
  print("F-score: {}".format(f1))

testoutput(rows)
```

```
Total no. of rows: 811
No Hate: 473
Actual no hate: 280
Weak Hate: 208
Actual weak hate: 51
Strong Hate: 130
Actual strong hate: 63
Precision: 0.48581997533908755
Recall: 0.9184149184149184
F-score: 0.635483870967742
```

Exporting results into results.csv

```
import csv

fields = ['Unique ID', 'Post', 'Actual Labels Set', 'Total Score', 'Result Hate Label' , 'Res
with open("/content/drive/MyDrive/Projects/NLP Project/project/results.csv", 'w') as csvfile:
    csvwriter = csv.writer(csvfile)

    csvwriter.writerow(fields)

    csvwriter.writerows(rows)
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = fields
x.add_rows(rows)
x.align = "l"
print(x[0:5])
```

```
+-----------+---------------------------------------------------------------------
| Unique ID | Post
+-----------+---------------------------------------------------------------------
| 1         | दृढ़ इच्छा शक्ति से परिपूर्ण प्रणबदा के लिए देशहित सर्वोच्च रहा।
|           |
|           | उनका निधन हम सब के लिए अपूरणीय क्षति है।
|           | ईश्वर दिवंगत आत्मा को अपने श्रीचरणों में स्थान दें। शोक संतप्त परिजनों के प्रति संवेदनाएं।
|           | ऊं शांति!!!
| 2         | भारतीय जनता पार्टी rss वाले इतने गिरे हुए हैं जहां मैं रहती हूं वहां मेरी जासूसी  करा रहें है उ
| 3         | कोरोना से निपटने की तैयारी / दिल्ली में 10 हजार बेड वाला दुनिया का सबसे बड़ा कोविड केयर
|           | https://t.co/9rlQowAsFh #Delhi @ArvindKejriwal  @rajnathsingh @AmitShah @D
| 4         | गवर्नर कॉन्फ्रेंस में PM मोदी बोले- शिक्षा नीति में सरकार का दखल कम होना चाहिए
|           | https://t.co/ZvKgxk6dbd
| 5         | यूपी: गाजीपुर में Toilet घोटाला, प्रधान व सचिव ने किया लाखों का गबन, मुर्दों के नाम पर ब
|           |
|           | #UP
|           | https://t.co/hxM1uNNmX2
+-----------+---------------------------------------------------------------------
```

Colab paid products - Cancel contracts here

Results:

We use a well defined criterion to segregate the tweets by the level of hatefulness:

- If the tweet has two or more words qualified as strongly negative, it is marked as strongly hateful. If it has one strongly negative word with non-zero hate verbs or non-zero theme-based nouns, it is marked as strongly hateful. If it has at least one word each from our hate-verbs lexicon and themed nouns, it is likewise marked as strongly hateful.

- If it has one word that is strongly negative, but no other word from our other lexicons, it is weakly hateful. If it contains one weakly hateful word with a theme based noun, it is weakly hateful. If it contains neither of the above but contains a hate-verb, it is also weakly hateful.

- If the tweet satisfies neither of these criteria, it is judged as non-hateful.

Now that we have defined the criteria for which tweets are judged hateful or otherwise, we can finally test the algorithm. To do this, we first need to manually define what the objective results for hatefulness must look like so that we may then compare them to the results obtained from our various algorithms. We do this by using the tags that are a part of the corpus, which has already been judged for hate speech. Instead of using the criteria the corpus uses (non-hostile, defamation, fake, hate, offensive), we have used a simpler criterion of no hate, weakly hateful and strongly hateful in our project. Hence, we need a connection between the two. For this, we consider that:

• All the non-hostile tweets correspond to “not hate”.
• All the tweets marked as fake or defamatory correspond to “weakly hateful”.
• All the tweets marked as hate, offensive or a combination of one or more tags.
(Which, as mentioned before, cannot contain non-hostile) are deemed to correspond to “strongly hateful”).

Therefore, we will consider the aforementioned criteria as the objectively correct standard to compare against when evaluating the precision (ratio of true positives to that of the total positives, i.e., true+false positives), recall (ratio of true positives to the sum of true positives and false negatives), and F-score (harmonic mean of precision and recall) of our algorithm. Let's now discuss the conclusions we reached. The findings we got with and without the subjectivity analysis are examined first, followed by the results we got when we took subjectivity into account. Also, as we had mentioned previously, we wished to check for the effect had on the final results by the various parameters used to quantify the hate speech, so each result table will have different rows that signify the parameters used while getting the result.

Feature sets	Precision (%)	Recall (%)	F-score (%)
Semantic (weakly/ strongly -ve)	39.33	77.99	52.29
Semantic + hate-verbs	39.35	78.03	52.32
Semantic + hate-verbs + nouns	42.54	84.35	56.55

As you can see, the hate-verbs by themselves do not greatly affect our model's accuracy, but the themed-noun lexicon significantly increases it.

Feature sets	Precision (%)	Recall (%)	F-score (%)
Semantic (weakly/ strongly -ve)	45.37	84.98	59.16
Semantic + hate-verbs	45.49	85.02	59.27
Semantic + hate-verbs + nouns	48.58	91.84	63.54

The fact that adding subjectivity hints significantly increases the F-score of our hate-speech detector (almost 7% increase) highlights the significance of looking for contextual nouns when looking for and identifying hate speech.

Conclusion:-

In this digital era, it is becoming more important than ever to identify hate speech in languages like Hindi with large user populations. In order to detect and quantify hate speech with some degree of accuracy, our programme searches for lexical patterns using a rule-based methodology. Our algorithm's examination of subjectivity served a crucial purpose in this regard, and by taking it into account, the precision of hate speech identification was much increased

References:

[1] arXiv:cs/0409058 [cs.CL]

[2] Michael Chau, Jennifer Xu,Mining communities and their relationships in blogs: A study of online hate groups,International Journal of Human-Computer Studies,Volume 65, Issue 1,2007,Pages 57-70,ISSN 1071-5819,https://doi.org/10.1016/j.ijhcs.2006.08.009.

[3] Y. Zhou, E. Reid, J. Qin, H. Chen and G. Lai, "US domestic extremist groups on the Web: link and content analysis," in IEEE Intelligent Systems, vol. 20, no. 5, pp. 44-51, Sept.-Oct. 2005, doi: 10.1109/MIS.2005.96.

[4] Mondal, Mainack & Silva, Leandro & Benevenuto, Fabrício. (2017). A Measurement Study of Hate Speech in Social Media. 85-94. 10.1145/3078714.3078723.

[5] Xu, Z., & Zhu, S. (2010). Filtering offensive language in online communities using grammatical relations. Paper presented at 7th Annual Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference, CEAS 2010, Redmond, WA, United States.

[6] https://www.researchgate.net/publication/262363934_Detecting_hate_speech_on_the_world_wide_web

[7] Dataset: https://github.com/mohit19014/Hindi-Hostility-Detection-CONSTRAINT-2021/blob/main/Dataset/valid.csv

[8] Hindi SentiWord: https://amitavadas.com/sentiwordnet.php