MONITORING AND PREDICITNG SOCIAL UNREST

Sai Srinivas Chetti
chettisa@buffalo.edu
UBID: 50418655

Vivek Reddy Ragireddy
vragired@buffalo.edu
UBID: 50419344

Shreya Reddy Ragireddy
sragired@buffalo.edu
UBID: 50419975

1 Abstract

Given an article/news, this real-world task introduces you to the task of identifying and predicting social unrest based on NLP and ML methods. This project contains 3 sub-tasks which aims to address this problem.
Information Extraction: Text data contains a lot of information but not all of it will be important to you. In this task we are going to tackle this problem by extraction important information from the whole corpus. The Information includes Event date, Source, Fatalities, Event type, Location etc. We incorporated a mix of rule-based methods and supervised learning methods. **Event date, Source, Fatalities, Event type, Location** etc. We incorporated a mix of rule-based methods and supervised learning methods.
Summary Generation: Given the structured data collected from the previous task, our goal here is to summarize the text using it. There are two concepts of summarization Extractive Summarization and Abstractive summarization. We leveraged Abstractive Summarization and use sequence to sequence models which is a special class of Recurrent Neural Networks architectures to achieve this.
Event Prediction from Text: Given a sequence of unrest events, we aim to predict the number of unrest events that occur in the future. We adopted regression approach for this task. We have developed a seq-to-seq Bi-directional LSTM model, the model takes previous 30-day wise unrests as an input and the model will predict the number of unrest events that occur each day in the next 30 days.

2 Introduction

Given an article or sentence, this real word task involves in Identifying and predicting social unrest. This task contains 3 additional subtasks:
1)Information Extraction: Extracting predefined keywords from text. We incorporated a mix of rule-based methods and supervised learning methods.
2)Summary Generation: Given a list of keywords, generate text and provide summary.
3)Event prediction from text: Given an article or text, predict event. We have developed a seq-to-seq Bi-directional LSTM model, given a sequence of date for n consecutive days, identity the number of total unrest events that occur in the next n dates.

3 Related Work

For task1, we referred to the spacey NER named entity extraction research papers to see how to extract keywords. Spacy provides an option to add arbitrary classes to entity recognition systems and update the model to even include the new examples apart from already defined entities within the model.
For task2, we referred to the **Publication: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer** that highlights the importance of transfer learning where a model is first pre-trained on a data rich task before being fine-tuned on a downstream task. The effectiveness has led to several approaches and practices. The encoder-decoder model has been pre-trained on a mixture of unsupervised and supervised tasks. They have achieved the state-of-the-art results on machine translation, summarization, question answering and many more use cases.
For task3, we referred to the **Publication: EMBERS AutoGSR: Automated Coding of Civil Unrest Events.** AutoGSR system is an ecosystem of filtering, ranking, and recommendation models to determine if an article reports a civil unrest event.
Data pre-processing: Fetch and clean performs keyword-based search to make sure that at least one protest related keyword is present in the text. Data Enrichment Comprises of several

data pre-processing techniques like named entity extraction, lemmatization, location identification etc.,

Models Ecosystem: applies several machine learning models on news articles to classify an incoming article into protest or non-protest.

4 Model Architecture

We have a total of three subtasks within the project. We will define the architecture of each task separately.

4.1 Task 1:Information Extraction:

In this task we were asked to extract 9 predefined keywords from the given text. We will explain below about the keywords and the methodologies behind extracting them.

Keyword 1: Fatalities

We have incorporated Spacy's NER Library for this task. Below are the steps involved.

1)**Data Preprocessing:** In this step we cleaned the given sentence by removing stop words, special characters, punctuations. Converted text to lower case. Then each sentence is split into tokens so that we can input them into the spacy's library.

2)**Extracting Cardinal's:** Pipelined the preprocessed sentences one by one into the spacy's library and extracted the string containing the cardinals.

3)**Processing results:** The results that we have is still very raw and contains a lot of extra information. We systematically tried to extract relevant information from the results.

a)Replaced the results of text which return empty cardinals with 0.

b)Replaced DATE to 0 from the cardinal's using regex. Removed words that contains '/' in them.

c)Converted all the text to lower case.

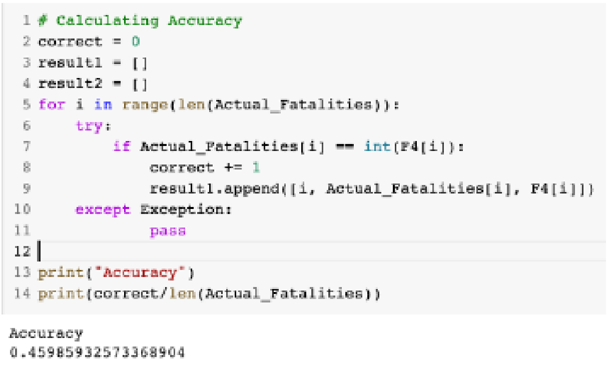
d) If a sentence contained more than 2 strings like 'at least twenty', we gain sent the cardinal to spacy and extracted only 'twenty'.

e) Converted cardinal text to numbers. eg: 'two' will be converted to 2

4)**Evaluating results:** We used the real validation data "Fatalities" to evaluate the results.

The model predicted correct fatalities for 46% of the data.

Keyword 2 and 3: Event Type and Sub Type



```
# Calculating Accuracy
correct = 0
result1 = []
result2 = []
for i in range(len(Actual_Fatalities)):
    try:
        if Actual_Fatalities[i] == int(F4[i]):
            correct += 1
            result1.append([i, Actual_Fatalities[i], F4[i]])
    except Exception:
            pass

print("Accuracy")
print(correct/len(Actual_Fatalities))
```

```
Accuracy
0.45985932573368904
```

Figure 1: Accuracy of Fatalities

We have applied classification approach to extract these keywords. Below are the steps involved.

1)**Data Preprocessing:**

This step contains multiple sub steps to it:

a)Removing null sentences and basic preprocessing steps mentioned in the above task.

b)Encoded and converted the Event Type (6 classes) and Event Sub Type(25 classes) classes to numbers.

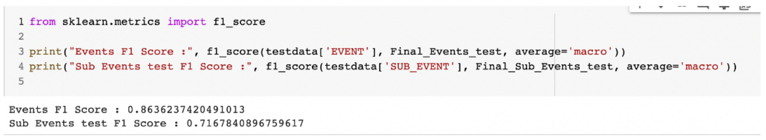
c)Tokenized the sentences using spacy's library.

2)**Text Encoding:** We need to convert the tokens in each sentence to numbers before feeding it for training the model. Used Bag of Words approach to create the vocabulary and encoded each sentence. Now the sentence is ready to be trained by the model with glove embeddings.

3)**Training:** We used Bi-directional LSTM with 2 layers to do the classification. We trained the model with every sentence from the training data. Used dropout regularization and linear activation function. Divided the data into batches with size 5000 and fed it into the model using Data Loader. Trained the model for 50 epochs with a learning rate of 0.01.

4)**Evaluating results:** After performing classification obtained the following results. Event Prediction, F1 score: 0.863

Sub Event Prediction, F1 score: 0.716



```
from sklearn.metrics import f1_score

print("Events F1 Score :", f1_score(testdata['EVENT'], Final_Events_test, average='macro'))
print("Sub Events test F1 Score :", f1_score(testdata['SUB_EVENT'], Final_Sub_Events_test, average='macro'))
```

```
Events F1 Score : 0.8636237420491013
Sub Events test F1 Score : 0.7167840896759617
```

Figure 2: F1 scores of Event Type and Event Sub Type

Keyword 4,5,6,7, 8: Actor1, Actor2, Inter1, Inter2 and Interaction:

For Actors, we have applied classification approach, we trained the model with training data and tested it with validation data. We have applied regression approach to predict rest of the keywords. Most of the steps are very similar to the above classification approach. Here instead of classification we used Uni-directional LSTM regression approach. Steps involves data preprocessing, encoding text, training and testing.

Keyword 9: Location

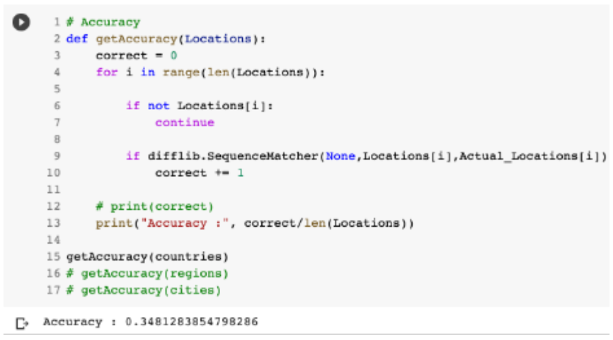
We have used spacy's NER and python iLocation-tagger libraries to extract location. Below are the steps involved.

1)Data preprocessing: Same steps mentioned in above approaches.

2)Extracting Locations: In the first step pipelined each sentence to ilocationtagger library to extract countries, states and cities. In the next step, sentences for which the above library did not generate and results, those sentences are passed on to spacy's ner library to extract locations entity.

3)Processing results: The results that we have is still very raw and contains a lot of extra information. We systematically tried to extract relevant information from the results. If a sentence has more than one location, concatenated those locations.

4)Results: Our model correctly extracted 35% of the locations correctly.



```
# Accuracy
def getAccuracy(Locations):
    correct = 0
    for i in range(len(Locations)):

        if not Locations[i]:
            continue

        if difflib.SequenceMatcher(None,Locations[i],Actual_Locations[i])
            correct += 1

    # print(correct)
    print("Accuracy :", correct/len(Locations))

getAccuracy(countries)
# getAccuracy(regions)
# getAccuracy(cities)
```

```
Accuracy : 0.3481283854798286
```

Figure 3: Accuracy of Locations

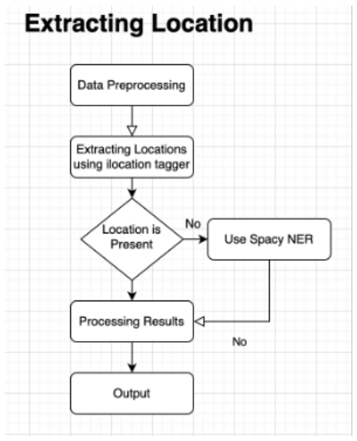


Figure 4: Architecture Diagram of Extracting Fatalities

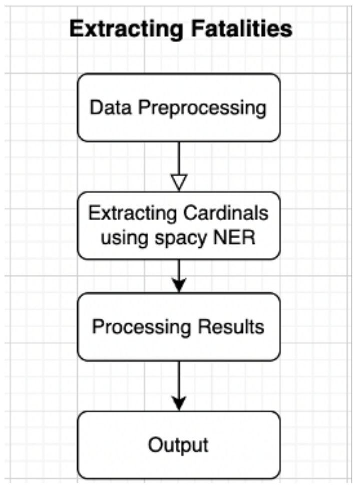


Figure 5: Architecture Diagram of Extracting Event Type and Sub Type

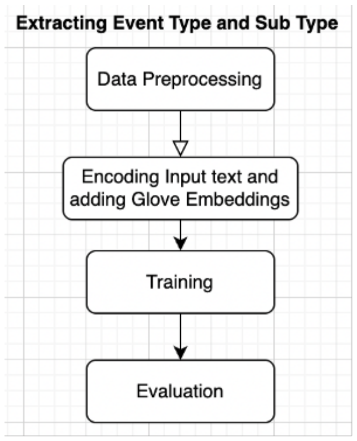

Extracting Event Type and Sub Type
Data Preprocessing
Encoding Input text and adding Glove Embeddings
Training
Evaluation

Figure 6: Architecture Diagram of Extracting Locations

4.2 Task2 :Summary Generation:

We utilized the idea of a new state of the art model, text-to-text transfer transformer provided by google (t5-base) is an encoder-decoder model pre-trained on multi-task mixture of unsupervised and supervised tasks that takes in structured data and generates a text, and it has the capability of taking combination of text and numerical data into the model that rights suits the ask for this task. The steps followed to achieve are

1)Data preprocessing: Removing spaces between words and identifying empty/no data columns in structured data and there by not passing the rows to our model. Further, we added input padding to give equal weight to every sequence that we pass to model. Moreover, we have removed nan values with few features in structured dataset.
2)Loading the preprocessed data and randomly shuffle the data to generalize the loss and then move the model to GPU in Colab.
3)The input sequence is fed to the model using input_ids (Encoded input sequence).. The target sequence is shifted to the right, i.e., prepended by a start-sequence token and fed to the decoder using the decoder_input_ids(decoded input sequence).
4)Used Adafactor optimizer with the recommended values for the t5-base model. Learning rate that best worked is 0.001.
5)For baseline, we explored many t5 models with different sizes as t5 comes in different sizes and as the size increases for the model it is implied that it has millions of parameters increase compared to previous ones.
6)Further, we tried to fine tune the model by tweaking Learning rate for Adafactor optimizer and reducing the batch size, epochs etc.

The evaluation metrics used for this problem is perplexity and rouge scores. They are shown in below results.

Architecture Diagarams:

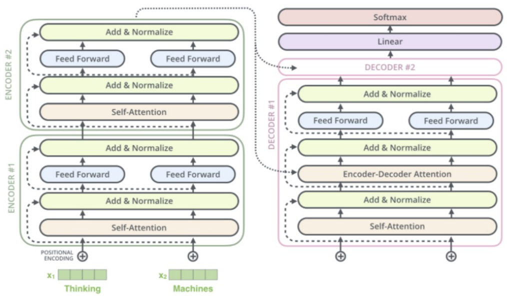


Figure 7: Architecture Diagram for Task-2

4.3 Task3: Event Prediction from text

Approach1 We have developed a seq-to-seq Multi-step Bi-directional LSTM time series model, given a sequence of date for n consecutive days,

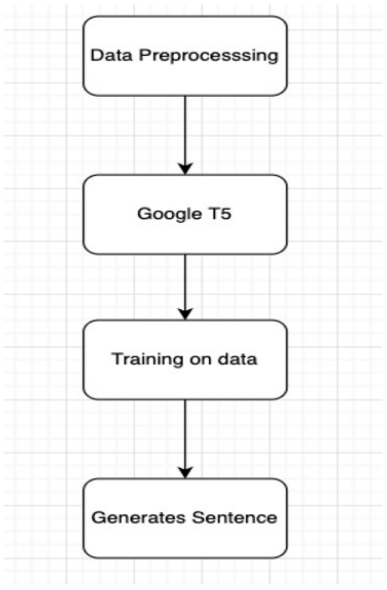
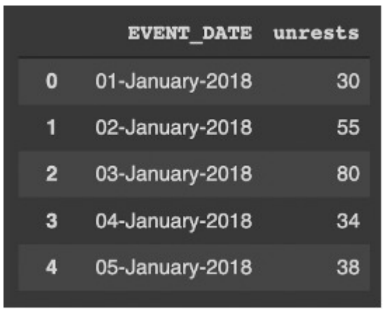


Figure 8: Flow Diagram for Task-2

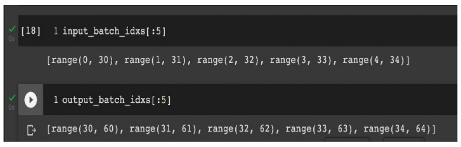
identity the number of total unrest events that occur in the next n dates. Below are the steps involved. Below we can see in the image how the model gets trained with input and output sequence of length 30. The first training sequence is input sequence is between (0,30) and corresponding indexes in out sequence (30, 60).
1. Data Preprocessing: Same steps mentioned in above approaches.
2. Calculating total numbers of unrest events for a given date and storing it as the target variable. The preprocessed data (image below) contains the respective dates and correcting number of unrest events.



	EVENT_DATE	unrests
0	01-January-2018	30
1	02-January-2018	55
2	03-January-2018	80
3	04-January-2018	34
4	05-January-2018	38

Figure 9: Input data

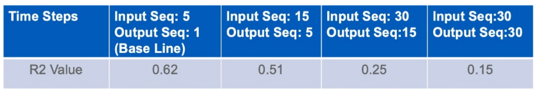
3. Training:
a. We used Bi-directional LSTM with 2 layers to do the regression. We trained the model with every sentence from the training data. Used dropout regularization and linear activation function. Divided the data into batches with size 10 and fed it into the



```
[18]  1 input_batch_idxs[:5]

      [range(0, 30), range(1, 31), range(2, 32), range(3, 33), range(4, 34)]

      1 output_batch_idxs[:5]

      [range(30, 60), range(31, 61), range(32, 62), range(33, 63), range(34, 64)]
```

Figure 10: Input data ids

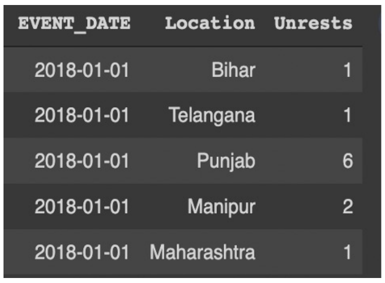
model using Data Loader. Trained the model for 200 epochs with a learning rate of 0.01. Divided the data into 8020% test data.
b. Developed multiple models that takes input sequence of length n and provides output sequence of length n. Used encode- decoder architecture.
1. Input Sequence: 5, Output Sequence: 1
2. Input Sequence: 15, Output Sequence: 5
3. Input Sequence: 30, Output Sequence: 15
4. Input Sequence: 30, Output Sequence: 30
4. Evaluating Results: Below are the R-Squared Values for these models when tested on test data.



Time Steps	Input Seq: 5 Output Seq: 1 (Base Line)	Input Seq: 15 Output Seq: 5	Input Seq: 30 Output Seq:15	Input Seq:30 Output Seq:30
R2 Value	0.62	0.51	0.25	0.15

Figure 11: Results

Approach2 We have also tried a different approach to include location during training of the model, previously we considered only the number of unrests, but here we trained the model with the location attribute to get location wise predictions. The predictions for 2 sequences are shown below for input sequence 15 and output sequence 5. Here in the table the first column represents number of unrests, and the second column represents location codes.
After comparing this with the actual results we found that the results are not satisfactory.
Training data and predictions:



EVENT_DATE	Location	Unrests
2018-01-01	Bihar	1
2018-01-01	Telangana	1
2018-01-01	Punjab	6
2018-01-01	Manipur	2
2018-01-01	Maharashtra	1

Figure 12: Input data with location for Approach 2

Architecture Diagrams for Task-3

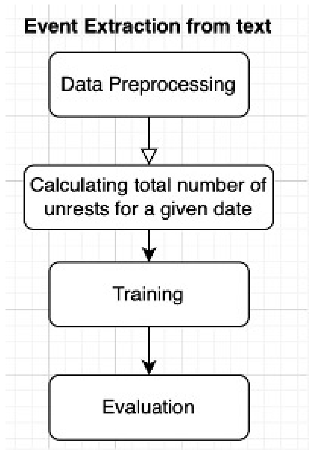

Event Extraction from text
Data Preprocessing
Calculating total number of unrests for a given date
Training
Evaluation

Figure 13: Flow diagram for Event extraction from text

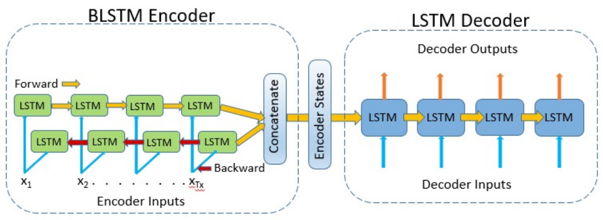
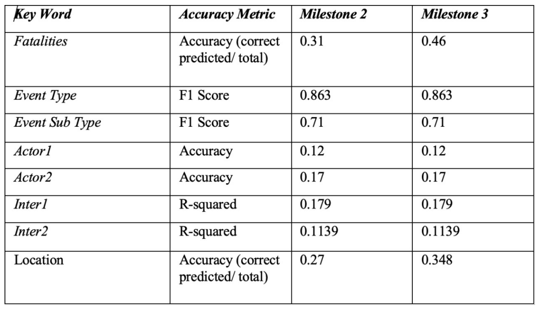


Figure 14: Architecture Diagram for Task-3 Approach 1

5 RESULTS:

Task1: Information Extraction



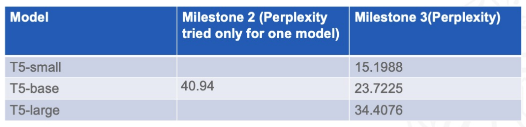
Key Word	*Accuracy Metric*	*Milestone 2*	*Milestone 3*
Fatalities	Accuracy (correct predicted/ total)	0.31	0.46
Event Type	F1 Score	0.863	0.863
Event Sub Type	F1 Score	0.71	0.71
Actor1	Accuracy	0.12	0.12
Actor2	Accuracy	0.17	0.17
Inter1	R-squared	0.179	0.179
Inter2	R-squared	0.1139	0.1139
Location	Accuracy (correct predicted/ total)	0.27	0.348

Figure 15: Task-1 Results

Task2:Summary Generation(In next page Fig 16 and Fig 17)

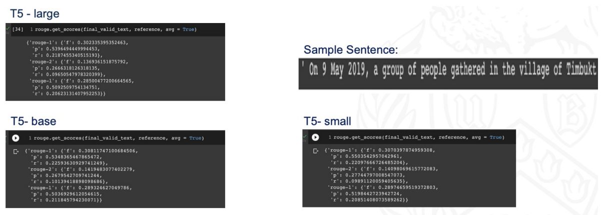
Task3:Event Prediction from text(In next page Fig 18 Fig19 Fig 20 Fig 21 and Fig 22)

Prediction plot comparing actual values and predicted values for different model architectures. For the baseline model with input sequence 5 and output sequence 1 the R2 suggests a good fit. When the input sequence is 15 and output is 5, the R2 is 0.51. We can observe that as the sequence gets increased, the R2 value is getting decreased. We think this is because of underfitting, since we have



Model	Milestone 2 (Perplexity tried only for one model)	Milestone 3(Perplexity)
T5-small		15.1988
T5-base	40.94	23.7225
T5-large		34.4076

Figure 16: Task-2 Results



T5 - large

```
[34]  1 rouge.get_scores(final_valid_text, reference, avg = True)

    {'rouge-1': {'f': 0.302335395352463,
      'p': 0.5396494449994453,
      'r': 0.2187455340515193},
     'rouge-2': {'f': 0.136936151875792,
      'p': 0.2666318126318135,
      'r': 0.09650547978320399},
     'rouge-l': {'f': 0.28500477200664565,
      'p': 0.5092509754134751,
      'r': 0.20623131407952253}}
```

Sample Sentence:

```
' On 9 May 2019, a group of people gathered in the village of Timbukt
```

T5- base

```
1 rouge.get_scores(final_valid_text, reference, avg = True)

{'rouge-1': {'f': 0.30811747100684506,
  'p': 0.5348365467865472,
  'r': 0.22593630929741249},
 'rouge-2': {'f': 0.1419483077402279,
  'p': 0.2679542709741244,
  'r': 0.10139418898098686},
 'rouge-l': {'f': 0.289324627049786,
  'p': 0.5036929612054615,
  'r': 0.211845794230071}}
```

T5- small

```
1 rouge.get_scores(final_valid_text, reference, avg = True)

{'rouge-1': {'f': 0.3070397874959308,
  'p': 0.5503542957042961,
  'r': 0.22097666726485204},
 'rouge-2': {'f': 0.14098069615772083,
  'p': 0.27744797008547073,
  'r': 0.09891120059405635},
 'rouge-l': {'f': 0.28974659519372803,
  'p': 0.5198442723942724,
  'r': 0.20851408073589262}}
```

Figure 17: Task-2 Results

only around 1480 data points and for longer sequences the model is getting underfitted. Same can be observed looking at the graphs.

6 Discussion and Error Analysis

Task1: Information Extraction

1)Keywords extracted using classification (event and sub event) gave better results compared to other keywords. And we don't think there will be any drastic improvement for these keywords' extraction.

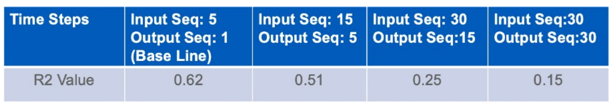
2)For the keyword Fatalities, using a basic approach of extracting cardinal entities from sentences gave much better results than we expected.

a)If a sentence has more than 1 cardinal, we tried to take the relevant one buy looking at the words around it and take the numbers closed to words like 'killed', 'murder' etc, but we haven't optimized in finding the relevant words, this can be area of improvement.

b)Also, if a sentence does not contain actual cardinal but a death can be written using words like "a person killed", our model does not identify these, this is another area of improvement.

3)For the keyword Actor1, Actor2, Inter1, Inter1 and Interaction the regression and classification approach did not provide us with satisfactory results. There is a lot of scope for improvement.

4)For Keyword Location, using a basic approach



Time Steps	Input Seq: 5 Output Seq: 1 (Base Line)	Input Seq: 15 Output Seq: 5	Input Seq: 30 Output Seq:15	Input Seq:30 Output Seq:30
R2 Value	0.62	0.51	0.25	0.15

Figure 18: Task-3 Results

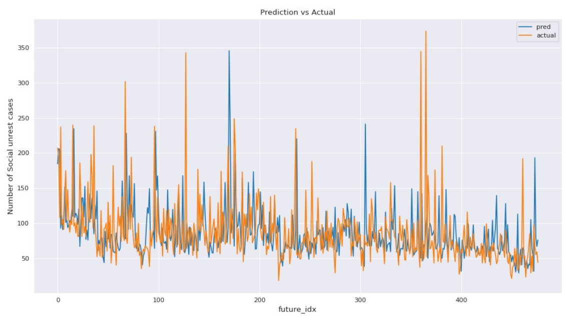


Figure 19: Seq 1 results for Task-3

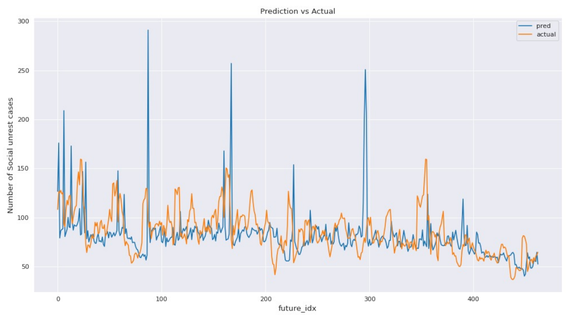


Figure 20: Seq 2 results for Task-3

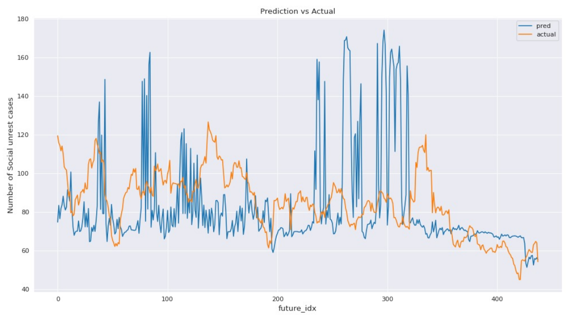


Figure 21: Seq 3 results for Task-3

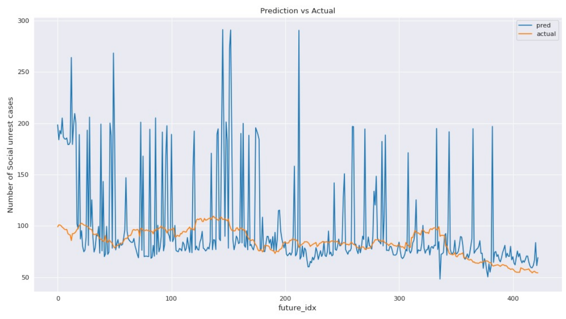
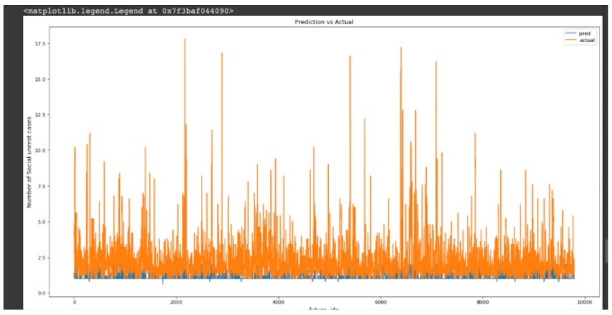


Figure 22: Seq 4 results for Task-3



```
<matplotlib.legend.Legend at 0x7f3baf044090>
```

Figure 23: Alternative approach graph which takes in locations too.

of extracting cardinal entities from sentences give an accuracy of 31%. A lot can be done to improve this baseline model like We have concatenated all the locations if a sentence has multiple locations in it, however this is brute force approach. We need to improve this step to extract only the relevant information, this can be done by training the model and converting this into a classification approach.

Task2: Summary Generation From the training perspective, we tried different possible learning rates for fine tuning the model with respect to adafactor, batch sizes and we found that learning rate 0.001 and batch size = 5 gives best results and with regards to improvement of the model maybe we could add additional data from social media, newspaper etc. to aid the summarization task.

Moreover, we can also fine tune the model with other transformer architectures and use a different and construct an efficient tokenizer based on Sentence Piece. For instance, we can take existing model and fine-tune on top of **Hugging Face's Transformers library**

Task3: Event Prediction from text For this task we trained the model, with 75% of the given data. We tested the model using the 25% test data for different input and output sequences. From the plot provided in "Results" section, we can see that the model does a fair prediction when the sequence is less, but as sequence length increases the model accuracy is getting reduced. However, a lot can be done to improve this model further like: 1)The poor result for longer sequences is due to the model getting underfitted. The performance can be improved if we train the model with more data. 2)Other areas of improvement are to do the regression using Transformers, which might give even better results. 3)Also, in our approaches to solve this task, we only included unrests column and location column as inputs for training. However, we can also include Notes into the training to help bring context to the sentences.

7 Conclusion

Task1: We incorporated a mix of rule-based methods and supervised learning methods, and we evaluated the model with macro F1 score. Further, we solved the problem using traditional machine learning algorithms and few fields were extracted using spacy NER

.

Task2: This was thought of as an inverse of task1, given different information fields from a reported event we leveraged the Google T5 model to generate an ACLED style summary of the occurred event. We evaluated this problem using perplexity and rouge scores. Perplexity is essentially an evaluation metric for language models and is exponentiation of cross entropy, whereas rouge computes rouge-1, rouge-2, and rouge-l where each represents consecutive matches of unigrams, bigrams and longest common subsequence with the generated text and the reference text that we have. It gives us a holistic view around the model.

Task3: We used Bi-directional LSTM with 2 layers to do the regression. We trained the model with every sentence from the training data. Used dropout regularization and linear activation function. Divided the data into batches with size 10 and fed it into the model using Data Loader. Trained the model for 200 epochs with a learning rate of 0.01. Divided the data into 80% train and 20% test data.

Developed multiple models that takes input sequence of length n and provides output sequence of length n. Used encode-decoder architecture.
1)Input Sequence: 5, Output Sequence: 1
2)Input Sequence: 15, Output Sequence: 5
3)Input Sequence: 30, Output Sequence: 15
4)Input Sequence: 30, Output Sequence: 30

8 Work Distribution

Tasks	Person in charge
Task1	Shreya Reddy
Task2	Vivek Reddy
Task3	Sai Srinivas Chetti

8.1 Bibliography

Task1:
https://spacy.io/api/
entityrecognizer
https://omdena.com/blog/
spacy-named-entity-recognition/
https://jovian.ai/aakanksha-ns/
lstm-multiclass-text-classification#
C52

Task2:
http://jalammar.github.io/
illustrated-transformer/
https://huggingface.co/docs/
transformers/model_doc/t5
https://towardsdatascience.com/
perplexity-intuition-and-derivation-105dd481c8f3
https://towardsdatascience.com/
the-ultimate-performance-metric-in-nlp-111df6c64460

Task3:
https://github.com/manohar029/
TimeSeries-Seq2Seq-deepLSTMs-Keras/
blob/master/Keras_Enc-Dec_
MinTempMel.ipynb
https://pangkh98.medium.com/
multi-step-multivariate-time-series-forecasting-using-lstm-92c6d22cd9c2
https://www.analyticsvidhya.
com/blog/2020/10/
multivariate-multi-step-time-series-forecasting-using-stacked-lstm-sequence-to-
https://www.adamsmith.
haus/python/answers/
how-to-calculate-r-squared-with-numpy-in-python