**Task 3**

1. High Level Idea

We are using Deep Learning approach to classify the text fragments for propaganda. The general approach is as follow:

1. Split article per sentences

2. Tokenize each word in the sentences

3. Feed tokens in a sequential model by using retrained word embedding model and get prediction for each token

4. Consolidate results on article level

The pros of this approach is that there are a lot of standard LSTM and CNN models for sentiment analysis of a text.

Main challenges are coming from the data prep. On the pre-modeling prep converting text to tokens keeping their relative positions and outcome. This is crucial for cases when tokenizer is droping words of the sentence. On the post-modeling prep converting back results to text positions.

1. Considered scenarios

We considered two scenarios for addressing the approach described above.

1. End-To-End modelling

There are 18 kinds of propaganda, as follow:

* Appeal\_to\_Authority
* Appeal\_to\_fear-prejudice
* Bandwagon
* Black-and-White\_Fallacy
* Causal\_Oversimplification
* Doubt
* Exaggeration,Minimisation
* Flag-Waving
* Loaded\_Language
* Name\_Calling,Labeling
* Obfuscation,Intentional\_Vagueness,Confusion
* Red\_Herring
* Reductio\_ad\_hitlerum
* Repetition
* Slogans
* Straw\_Men
* Thought-terminating\_Cliches
* Whataboutism

One idea is to use a multicatgorical classification for each token. There are 19 categories - 18 propagandas and non-propaganda. Unfortunately, there is overlapping between the propaganda fragments, which means that some tokens could belong to a several categories simultaneously.

Therefore, we decided to run a classification for each kind of Proganda. Unfortunately, as could be seen from the table below, not all of the propagandas are well populated.

|  |  |
| --- | --- |
| **Propaganda Type** | **#** |
| Loaded\_Language | 1627 |
| Name\_Calling,Labeling | 839 |
| Repetition | 427 |
| Doubt | 359 |
| Exaggeration,Minimisation | 350 |
| Flag-Waving | 182 |
| Appeal\_to\_fear-prejudice | 162 |
| Causal\_Oversimplification | 133 |
| Slogans | 110 |
| Black-and-White\_Fallacy | 83 |
| Appeal\_to\_Authority | 81 |
| Thought-terminating\_Cliches | 57 |
| Whataboutism | 52 |
| Reductio\_ad\_hitlerum | 35 |
| Reductio\_ad\_hitlerum | 35 |
| Red\_Herring | 18 |
| Obfuscation,Intentional\_Vagueness,Confusion | 9 |
| Straw\_Men | 8 |
| Bandwagon | 7 |

Practically, we received a good models just for top 2 propagandas.

1. Two Stage modelling

In order to be able to cover all kind of propaganda, we split the modelling tasks on 2 phases.

Phase 1. Detect a propagandistic phrase.

Phase 2. Classify a propagandistic phrase.

4.1 Word embedding

We used a pretrained Glove model on Wikipedia corpus having 400k words in it. We tested different size of the vectors - 50, 100, 200 and 300. Based on the test we selected word embedding of size 200.

4.2 Final model used

On training side we considered two approaches - Bidirectional LSTM models and 1D CNN models. We selected CNN models, because LSTM models are much slower on the hardware we used, therefore it took much more time for selecting the appropriate network architecture and hyper parameters

4.2.1 Propaganda identification

model = Sequential()

model.add(Embedding(num\_words, EMBEDDING\_DIM, weights=[embedding\_matrix], trainable=False))

model.add(Conv1D(filters=LATENT\_DIM, kernel\_size=5, padding="same"))

model.add(MaxPooling1D(pool\_size=3, strides=1, padding="same"))

model.add(Conv1D(filters=LATENT\_DIM, kernel\_size=4, padding="same"))

model.add(MaxPooling1D(pool\_size=4, strides=1, padding="same"))

model.add(Conv1D(filters=LATENT\_DIM, kernel\_size=3, padding="same"))

model.add(MaxPooling1D(pool\_size=5, strides=1, padding="same"))

model.add(TimeDistributed(Dense(20, activation="relu")))

model.add(Dense(1, activation="sigmoid"))

model.compile(

loss='binary\_crossentropy',

optimizer=Adam(lr=0.01),

metrics=['accuracy']

)

Layer (type) Output Shape Param #

=================================================================

embedding\_12 (Embedding) (None, None, 200) 3786200

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv1d\_19 (Conv1D) (None, None, 32) 32032

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling1d\_15 (MaxPooling (None, None, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv1d\_20 (Conv1D) (None, None, 32) 4128

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling1d\_16 (MaxPooling (None, None, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv1d\_21 (Conv1D) (None, None, 32) 3104

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling1d\_17 (MaxPooling (None, None, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

time\_distributed\_10 (TimeDis (None, None, 20) 660

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_20 (Dense) (None, None, 1) 21

=================================================================

Total params: 3,826,145

Trainable params: 39,945

Non-trainable params: 3,786,200

F1 score on the train-dev set is 0.25

4.2.2 Propaganda classification

model = Sequential()

model.add(Embedding(num\_words, EMBEDDING\_DIM, weights=[embedding\_matrix], trainable=False))

model.add(Conv1D(filters=LATENT\_DIM, kernel\_size=5, padding="same"))

model.add(MaxPooling1D(pool\_size=3, strides=1, padding="same"))

model.add(Conv1D(filters=LATENT\_DIM, kernel\_size=4, padding="same"))

model.add(MaxPooling1D(pool\_size=4, strides=1, padding="same"))

model.add(Conv1D(filters=LATENT\_DIM, kernel\_size=3, padding="same"))

model.add(GlobalMaxPool1D())

model.add(Dropout(0.2))

model.add(Dense(128, activation="relu"))

model.add(Dropout(0.2))

model.add(Dense(18, activation="softmax"))

model.compile(

loss='binary\_crossentropy',

optimizer=Adam(lr=0.01),

metrics=['accuracy']

)

Layer (type) Output Shape Param #

=================================================================

embedding\_15 (Embedding) (None, None, 200) 1359600

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv1d\_7 (Conv1D) (None, None, 32) 32032

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling1d\_3 (MaxPooling1 (None, None, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv1d\_8 (Conv1D) (None, None, 32) 4128

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling1d\_4 (MaxPooling1 (None, None, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv1d\_9 (Conv1D) (None, None, 32) 3104

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

global\_max\_pooling1d\_10 (Glo (None, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_19 (Dropout) (None, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_19 (Dense) (None, 128) 4224

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_20 (Dropout) (None, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_20 (Dense) (None, 18) 2322

=================================================================

Total params: 1,405,410

Trainable params: 45,810

Non-trainable params: 1,359,600

F1 score on the train-dev set is 0.35

1. Options for further research

During the brainstorming sessions, there were 2 additional ideas, which we did not have time to work on.

1. Based task 3 on the results of task2. As result all non-propagandistics sentences could be filtered and taks 3 will focus just on finding the propagandistic phrase in a sentence which is propaganda.

2. To replicate YOLO object detection model for this task.