understanding the public opinion
on
legalization of abortion
on
Twitter.

Project in AI
236502
faculty of computer science, Technion

Reham Farhat
208777045
reham.farhat@campus.technion.ac.il

Yousef Tannous
206017147
youseft@campus.technion.ac.il

Introduction

The abortion debate has long been one of the most controversial topics in the world. Ethical, moral, philosophical, biological, religious, and legal issues surrounding abortion are related to value systems. Opinions of abortion mainly revolved about fetal rights (pro-life), women's rights (pro-choice), and governmental authority.

With public opinion so divided, changing the abortion laws through legislative action can erupt popular anger and a wave of mass protests, the recent wave lasted 3 months in Poland known as the October–December 2020 Polish protests and it was the biggest protest in the country since 1989.

Therefore, we cannot ignore the fact that abortion is also a political issue, candidates' opinions on the laws of abortion could turn off voters and cost them the election, that might be why Trump, U.S president, has changed his view on abortion, going from pro-choice in 1999 to pro-life with exceptions as he planned a presidential run.

Thus, understanding the public opinion might be crucial for developing the abortion ground and stating new laws that correspond with the public opinion.

While polls are the traditional method of understanding public opinion, in practice pollsters need to balance the cost of a large sample against the reduction in sampling errors, using this way force the pollsters to ask the participants directly, you may reveal your identity, waste a lot of time and money, people may change their mind or might not have the time to answer the poll’s questions, so to get complete responses, it might be necessary to include more participants.

Twitter has been widely used as a platform for sharing opinions and an instrument for political engagement. Studying the public opinion on abortion and the reasons that correlate with it using Twitter can indicate what is the distribution of opinions. So, we went to different deep learning and sentiment analysis algorithms applying them to Twitter’s data.

Proposed solution

In our problem stands the difficulties of stance detection, where the sentiment is not generic but with respect to a specific topic.
We propose RNN based deep neural network within a two-phase architecture,
In the first phase, we classify the subjectivity (positive \ negative \ neutral) with respect to our topic. In the second phase, we classify the subjective input as to whether it has a stance towards the topic (favor \ against \neutral) using the outputs of the first phase.
After we train our model to detect stances from tweets with respect to our topic, we will show the distribution of the tweets on a heat map to get a wider perspective of how the public is divided.

Data extraction and analysis

We used the training and testing datasets provided by the semeval 2016 stance detection task, which contains labeled tweets with respect to different topics including the legalization of abortion. For improved versions of our model we used pre-trained embeddings using the GloVe word2vec data and added positional information for the attention layers. For tuning the hyperparameters of the model we have conducted several experiments with the data, see **experiments**.
After we train and test our model, we will extract tweets using that are related to our topic (by searching for keywords and hashtags) twitter's API, classify them using our model and display their distribution on a heat map.
We use pytorch for programming and as a library for deep learning models, pycharm as the IDE, anaconda, and jupyter for running code and experiments.

RNNs

In the context of learning from **sequences** of inputs, RNNs are capable of learning a transformation of one sequence into another and has time-dependent modeling that the later outputs are influenced by early inputs through hidden states, which also serves in saving a state while "reading" a sentence to comprehend complicated sentences and extract their sentiment.
LSTMs (Long-short term memory) are over time versions of the RNNs and their main idea is to have "soft-gates" that control how much of the previous state affects the next state relative to the proposed next state.
In this project, we use the LSTMs to extract the sentiment of a given sequence in the first phase, and then we use it to extract a stance from the transformed sequence of the input.

Encoder-Decoder architecture

A common architecture used in many tasks is the encoder-decoder pattern. The encoder maps the input to some latent representation, usually of a low dimension. The decoder applies a different mapping, from the latent space to some other space.
In this project, our “Encoder-Decoder” pattern is found in the two phases of the architecture.

Attention

In deep learning contexts, attention is a term used for a family of related mechanisms which, in general, learn to predict some probability distribution over a sequence of elements. Intuitively, this allows a model to "pay more attention" to elements from the sequence which get a higher probability weight.
In this project we perform multi-head attention once which is described by the following formulas:

$$MultiHead(Q,K,V) = Concat(head1,\dots,headh)W$$

$$where\ headi = Attention(QWQ\ i, KWK\ i, V\ WV\ i)$$

In our project, the attention would be applied to the encoder’s hidden states output sequences which hold the information of the sentiment, and by practice, we will get more attention on words that most contribute to the sentiment classification of the sentence which indeed then helps in the second phase of the model to detect a stance with respect to the specific topic.

Using Attention for Alignment and Context

More specifically, we can consider the previous decoder state as the query vector, and the encoder hidden states as key and value vectors. The output is a weighted average of the value vectors, where the weights are determined by the compatibility function between the query and the keys, which intuitively gives context and helps align the stance to the subjective different parts of the sentence. The approach here is based on the classic paper "Neural Machine Translation by Jointly Learning to Align and Translate" by Bahdanau et al. (ICLR, 2015).

Positional Information Techniques

The attention mechanism is completely invariant to sequence order, thus we need to incorporate positional information so our transformed sequence which then would be used as an input to the second RNN to detect a stance would still hold positional information of the sequence.
One of the techniques that have been proposed to solve this problem is adding sinusoidal positional encodings (PE) vectors that are created using sine and cosine functions of different frequencies and then are added to the input embeddings.
The following sine and cosine functions are used:

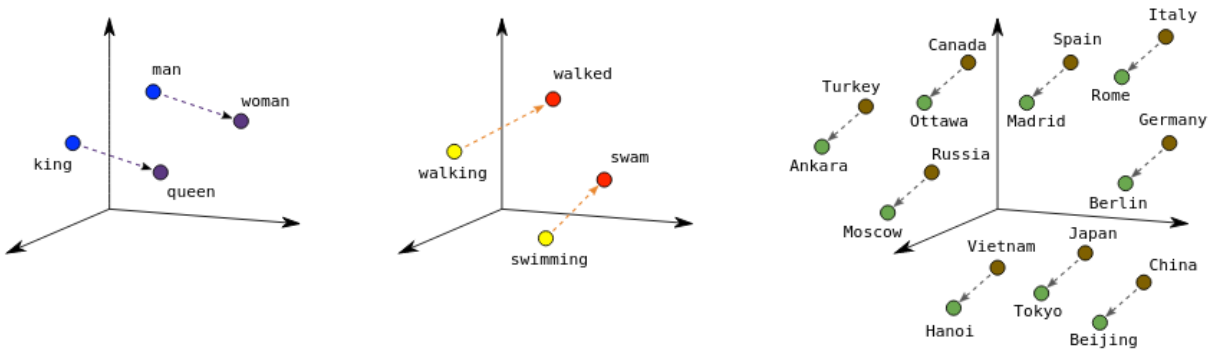
$$PE(pos, 2i) = sin(pos\ /\ (1000^{2i\ /\ dmodel}))$$

$$PE(pos, 2i+1) = sin(pos\ /\ (10000^{2i+1\ /\ dmodel}))$$

where pos is the sentence position and i is the dimension.

Embeddings

Word embedding transformation: words are projected to a dense vector space, where the semantic distance between words are preserved



Male-Female Verb Tense Country-Capital

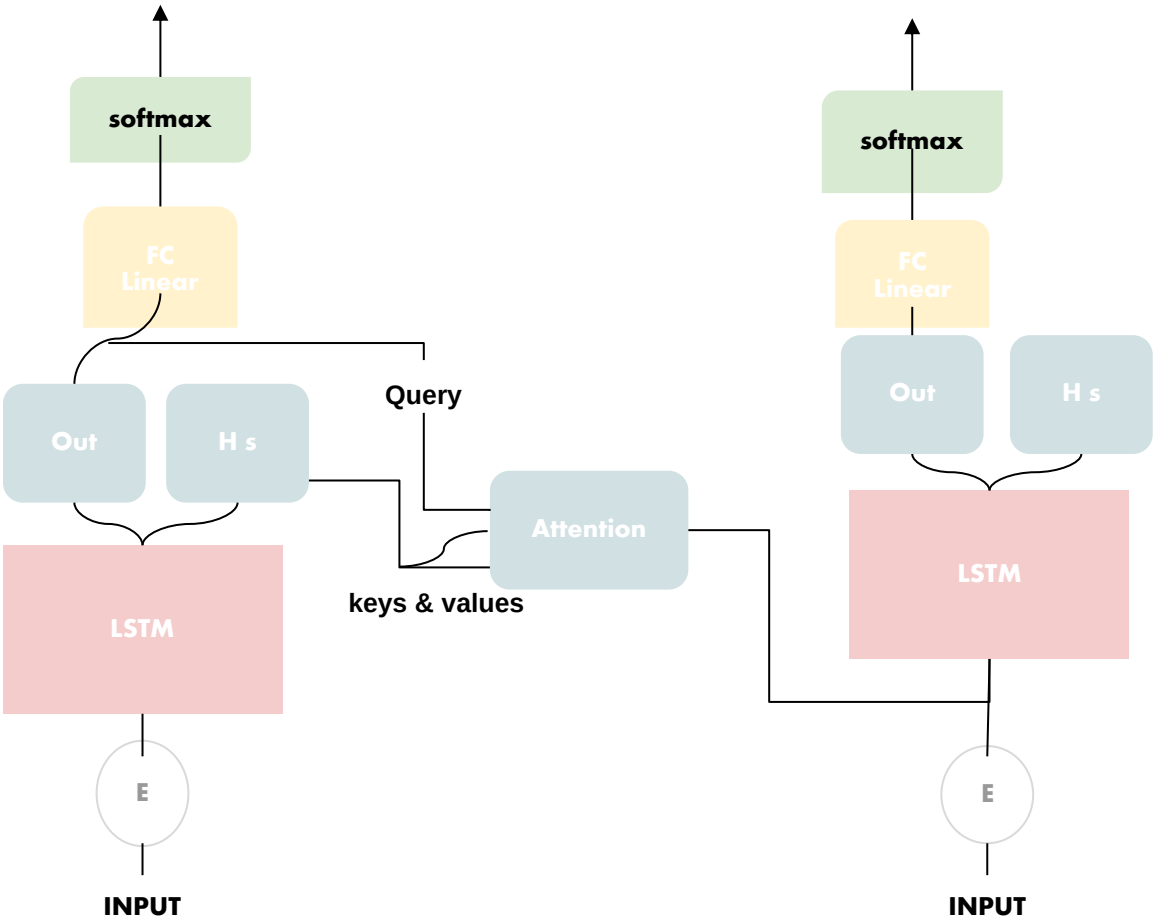
https://developers.google.com/machine-learning/crash-course/images/linear-relationships.svg

By default, the embedding vectors are randomly initialized, then will gradually be improved during the training phase, with the optimizer at each step, so that similar words or words in the same lexical field or with common stem will end up close in terms of distance in the new vector space.
Pre-trained embeddings are public embeddings that have been already trained on large datasets so instead of initializing our neural network weights randomly, we will use pre-trained embeddings as initialization weights.
We will show the efficiency of using pre-trained word embeddings as we use them in our improved model.
We will use the GloVe twitter 27b 200d embeddings as our initial weights, as we found that it helped accelerate the training and boost the performance of our model.

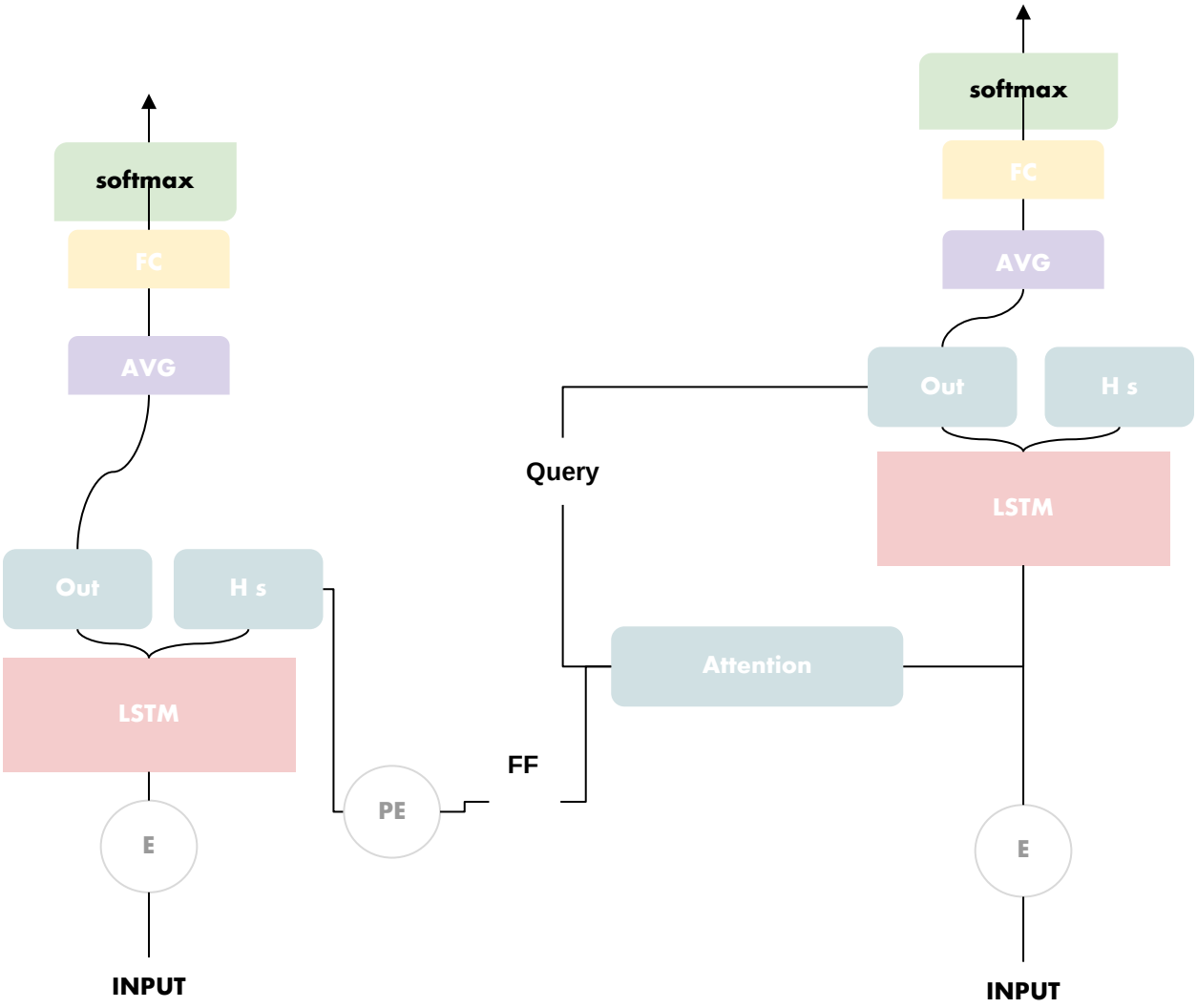
Proposed Architectures

We propose two-phase architecture models which are based on the encoder-decoder pattern, with an approach of generating outputs in two phases and training the model simultaneously to classify subjectivity (the output of the first phase) and to detect stance (the output of the second phase). the hidden states of the first phase are then passed on to a layer of attention and the outputs of the attention layer are used as additional information to the input of the second phase, intuitively subjective sentences could be an indication of having a stance towards the discussed topic, so we assure that our model “pays more attention” to subjective words and thus making a more “informed” transformation of the sequence for stance detection.

In our first attempt to implement the above architecture we have in our first phase a layer of embedding (not pre-trained), a 2-layer LSTM, a fully connected linear of dimension 3 layer for getting scores over the 3 classification labels, and a log softmax to normalize the output of the network to a probability distribution (see the figure below). In the second phase we have a layer of embedding (not pre-trained), attention layer, 2-layer LSTM, a fully connected linear dimension 3 layers, and a log softmax (see the figure below).



For the attention layer here we apply a self-attention mechanism without positional information, this is done by creating three vectors (query, key, value) for each sequence position, and then applying the attention mechanism for each position as a result each position of the transformed sequence incorporates the information of how each position relates to other positions in the input and so we use this information to guide the second phase to pay more attention to positions that most contribute to the subjectivity of the sentence.
In our second attempt to implement the architecture we intended to create the whole architecture closer to the encoder-decoder pattern, incorporating positional information for the attention layer and using pre-trained word2vec embeddings and global average pooling for gaining a sentence representation of fixed dimension.
In the first phase, we have an embedding layer (with pre-trained weights), LSTM layer, average layer, Linear layer of dimension 3, and a log softmax. In the second phase we have an embedding layer (with pre-trained weights), a Positional Encoder layer which is applied to the keys and values to gain positional information, feed-forward layer to obtain the Q, K, V on which the attention is performed, Attention layer, LSTM, average, Linear layer, and a log softmax (see figure below).



This time we had the approach of the encoder-decoder pattern that is used in seq2seq transformation models. We considered the previous decoder state as the query vector, and the encoder hidden states as key and value vectors.
for the models with the additional features incorporated we add a fully connected layer in the output followed by a softmax

Experiments

We had a standard approach for training where we split the input into batches and trained the model for the number of epochs. We defined our loss function as the addition of the two losses (the losses are computed in the first and second phases). We used the CrossEntropy loss functions for both of the losses. In addition to the two losses, we assure that our model stabilizes its training between the classification of the subjectivity and the detection of the stance, as a result, we get more subjectively informed stance predictions. We used the Adam optimizer. In our experiments, we intend to find the right parameters that would bring stabilized training, prevent overfitting and result in high accuracy which is defined as the correct predictions compared to the ground truth on the test set.

hyperparameters in our experiments:

- dropout: which for some probability disconnects layers in the model preventing our model from overfitting while training
- batch size: the number of batches we split our input into.
- epochs: the number of epochs our model trains
- batches each epoch: number of max batches in each epoch
- learn rate: the rate in which we want the loss function to converge.
- embedding dimension: the dimension we use for embedding the meaning of the words
- hidden dimensions: the hidden dimensions of the RNN layers we have in our models
- number of layers: the number of layers in our RNNs

Incorporating additional features

To further understand what defines stances towards a specific topic, we decided to incorporate the location of the user who tweeted as an additional feature to our classification model, so we can observe if there's a correlation between location and opinion and therefore make a more informative classification.
In the training and test data files, each tweet has a tweet id, which is a unique value for the tweet, so we used this id and twitter API to extract the location of the users depending on the twitter id, which is a unique value per user, some of the users don't attach their location so we excluded them.
The user specifies the location as a text so he could write the same place in a different way and it may include emojis, so to avoid the emojis and the language case sensitivity in the location syntax and represent the same place in a unique way (example below), we used the geopy geocoder, and for each place we extracted the latitude and the longitude. That helped the model to be more accurate.

location	LAT	LON
NYC	40.7127281	-74.0060152
NEW YORK CITY	40.7127281	-74.0060152
New york city :)	40.7127281	-74.0060152

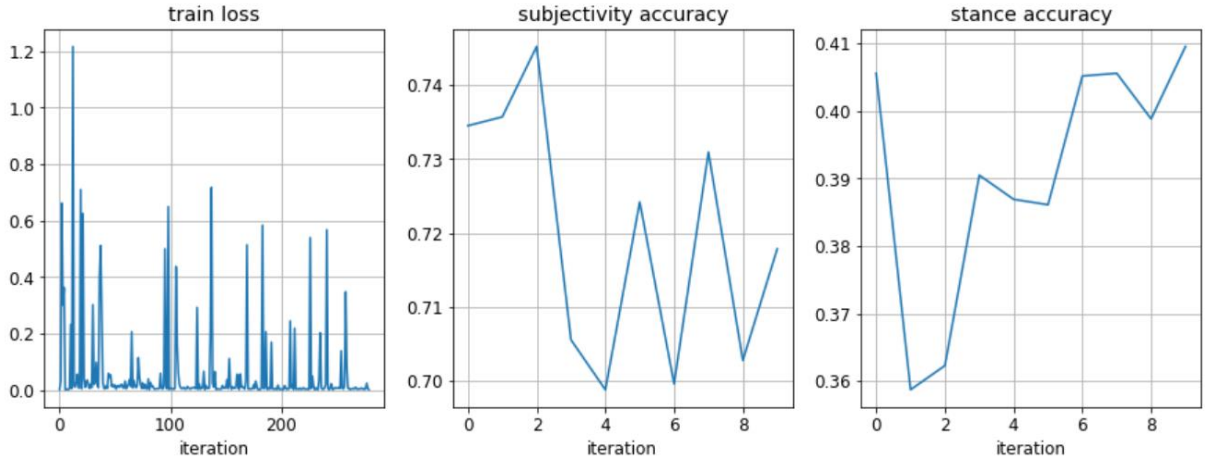
Baseline

We have implemented a baseline model based on LSTM which directly classifies the stance (one-phase) to have a comparative analysis. In the baseline model, we have a layer of embeddings (pre-trained), a layer of LSTM, averaging layer to gain a sentence representation, a linear layer, and a log softmax.
We had a similar approach for training and we trained the model on the same training set and evaluated it on the same test set. We conducted similar experiments to tune the hyperparameters of the model.

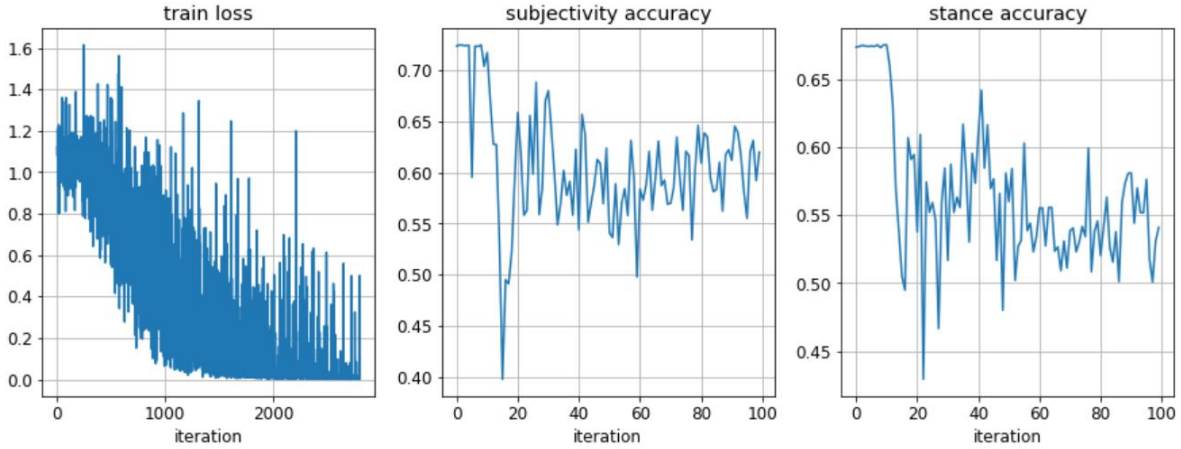
Results

We will display each of our evaluation graphs of the models we have implemented evaluated on the test set, then we compare in terms of accuracies and confusion metrics

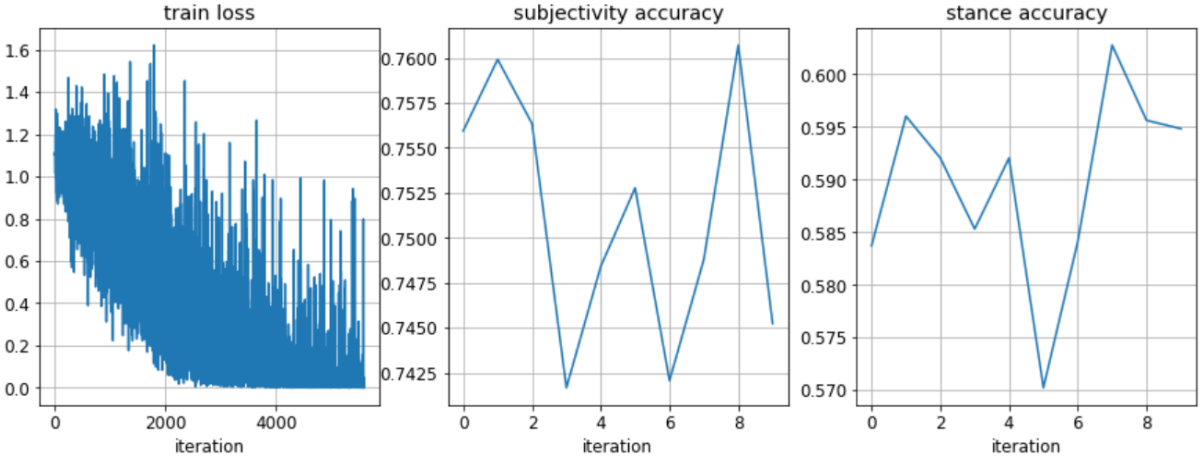
Our First architecture (see proposed architectures) not tuned:



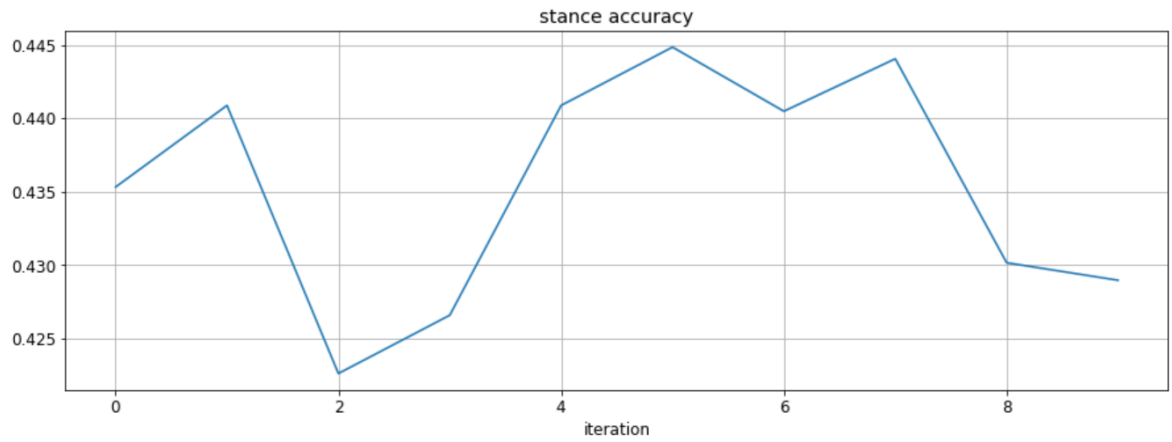
Our First architecture (see proposed architectures) tuned after experiments:



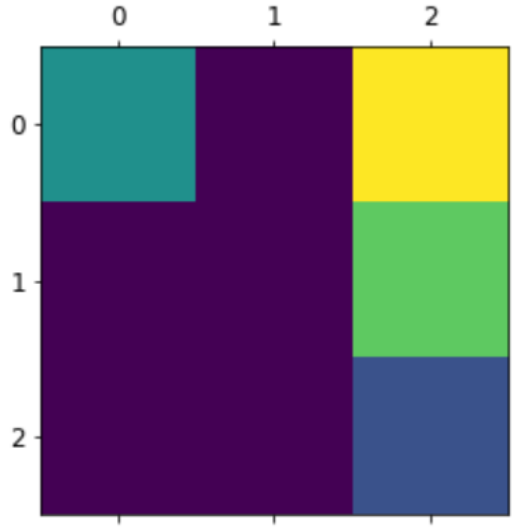
Our Second architecture (see proposed architectures) tuned after experiments:



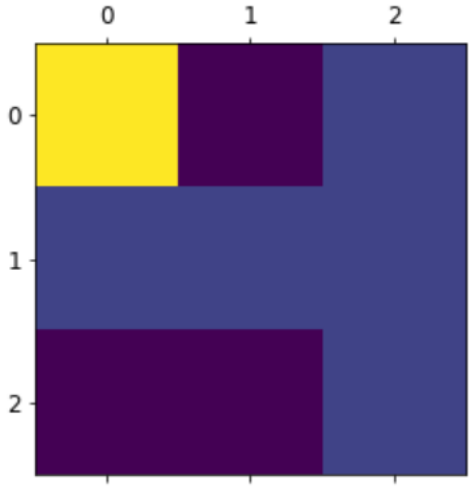
Our baseline architecture (see baseline) tuned after experiments:



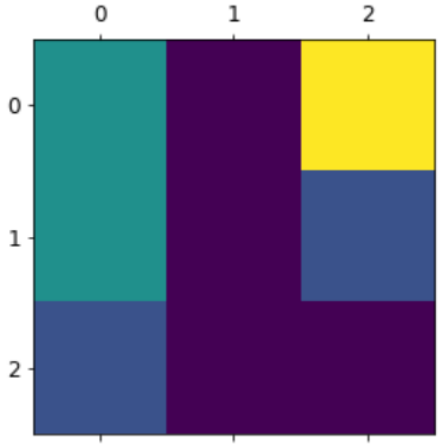
Confusion matrix of our first architecture on a single batch:
{'AGAINST': 0, 'NONE': 1, 'FAVOR': 2}



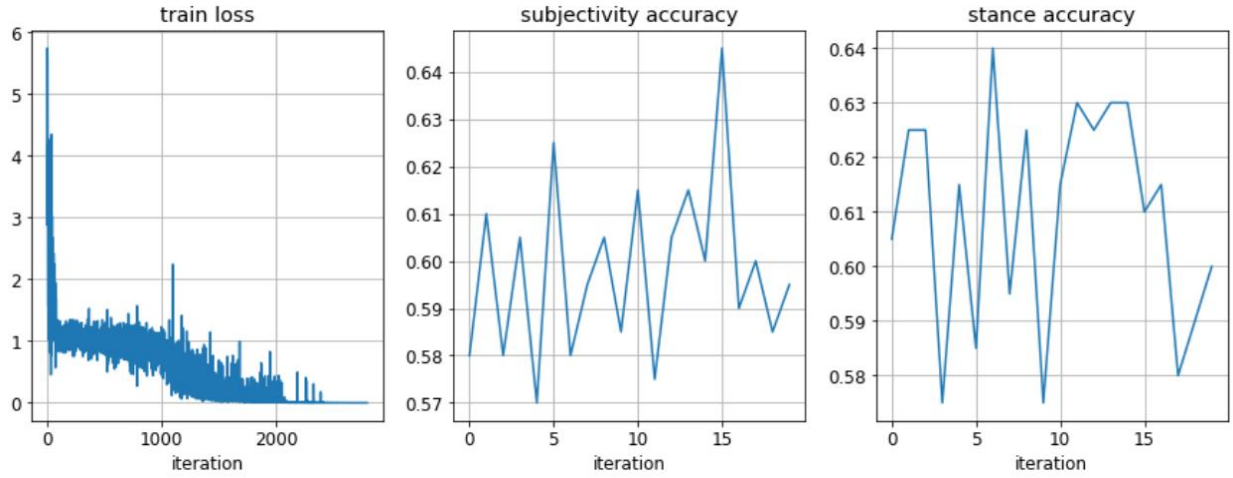
Confusion matrix of our second architecture on the same batch:
{'AGAINST': 0, 'NONE': 1, 'FAVOR': 2}



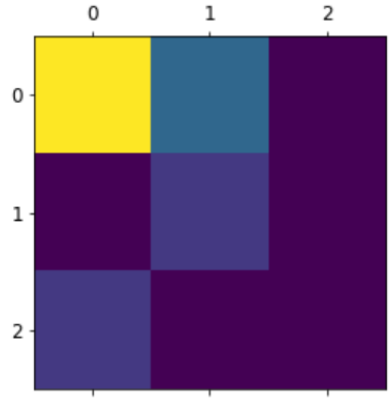
Confusion matrix of our baseline architecture on the same batch:
{'AGAINST': 0, 'NONE': 1, 'FAVOR': 2}



Results of the model with the location feature incorporated



Confusion matrix:



Analysis

As we can see in the results above our first architecture before tuning the hyperparameters had a poor accuracy in predicting the stance and as we conducted the experiments we understood that using 2-layers in the lstms, a higher dimension of embeddings, a lower dimension of the hidden states, lowering the learning rate to 1e-3, lowering the dropout parameter to 0.1 and batch size to 10 helped increasing its performance to 55%
(+10%) which also helped in better detecting non-neutral stance but yet we didn't see a good performance in detecting neutral sentences, we can also note that the decrease in the sentiment accuracy contributed to that poor performance.
In our second architecture adding the positional encoder and shifting towards the approach of alignment and context architecture showed better performance and higher accuracies, as we can see it increased the average accuracy of stance detection to about 61% and the sentiment prediction accuracy increased to 74%. The increase in the accuracy of the sentiment prediction resulted in better detection of neutral and non-neutral sentences.
Our results showed a generally better performance in comparison to the baseline model which is a one phase classification state of the art model for sequence classification.
We also observed that the second model we used had much more trainable parameters than the baseline model but way less trainable parameters than the first model due to the pre-trained embeddings which helped boost the training and performance.
We topped the accuracy performance of other models when we incorporated the location feature (%65) which indicates that there is a correlation between location and political view and incorporating that information in our data helps to make a more informative prediction and alleviates the accuracy.

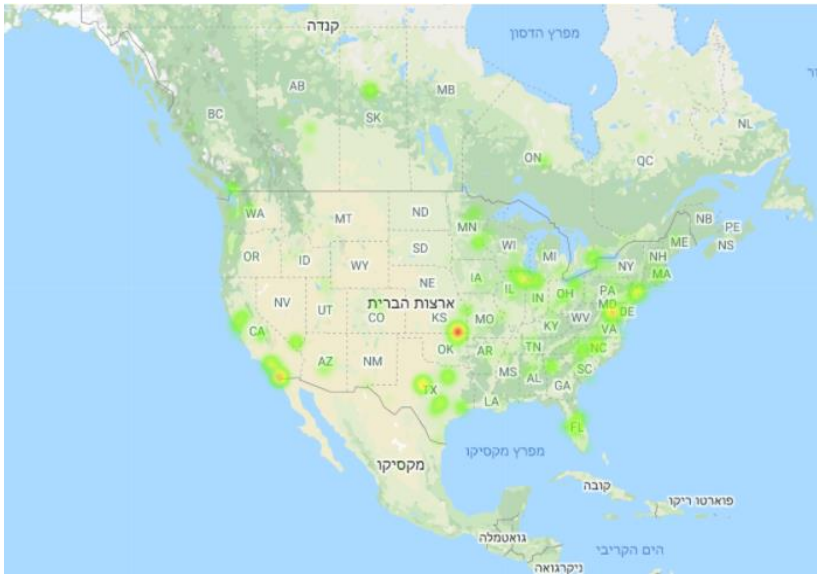
Maps

After we trained the models, we found that the last model – a two phase LSTM model that incorporates additional features -gives the highest accuracy.
in order to see how the public opinion is divided using this model, we extracted 1439 tweets related to abortion and we specified the language of the tweets to be English and the bounding box of the tweets location to be [-105.301758,39.964069 ,-105.178505,40.09455]). which is the bounding box of the US.
Tweet-specific location information falls into two general categories:

- Tweets with a specific latitude/longitude "Point" coordinate
- Tweets with a Twitter "Place.

So, we decided to extract according to the bounding box and show the result according to the "Place".
The following three heat maps show the distribution of the stance according to abortions in the USA.

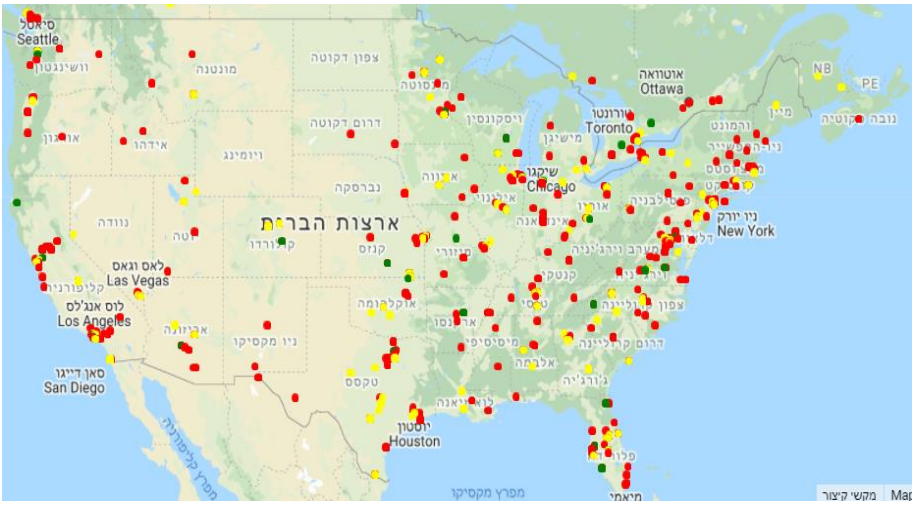
1. Pro-life stance



2. Pro-choice(support)

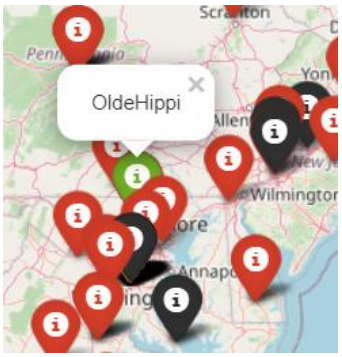
3. neutral stance

4 . one map for all stances, red for pro-life, green for pro-choice and yellow for neutral.



5. in the map we replace the yellow color to black for the neutral stance.

in the tweet.tsv file the tweet of the user OLDEHIPPI is "PRO life also mean getting vaccinate so you do spread the killer virus COVID ProLife GetVaccinated" and it was classified as a prolife.



while the user freedomFdtn's tweet "if education union really represent those who educate child they would align themselves with anything that help pay for abortion teacher union PlannedParenthood ohleg" was classified as anti-abortion.

we conclude from the above maps that this topic is highly debated in the world and specially in the united states and secondly, we get that the majority of the tweets are anti-abortion, this result was not surprising since that in our algorithm of extraction we can extract tweets up to 7 days ago using the twitter API, this limitation low the number of tweets we can extract.in addition, we limited the number of tweets for each of the hashtags up to 1000 , so since that we only get only 1439 tweets this means that there was no enough tweets for each of the hashtags, and in order to get more accurate and real result we have to use our algorithm for a period of time and collect more tweets as we can, the other reason can be the periodic trends of public opinion.
In addition, we see that the model can classify most of the tweets as against abortion, so we have to train it on more samples that support or have no stand toward abortion.The hashtags we used:

pro life hashtags:
#prolife, #abortionismurder, #prolifegeneration, #abortionban
, #antiabortion ,#abortionisevil ,#abortionpositive

Neutral hashtags:
#abortion,#abortions, #plannedparenthood

Prochoice hashtags:
#Prochoice, #abortionisnotmurder, #abortionisahumanright, #abortionrights, #mybodymychoice, #abortionishealthcare, #abortionisawomansright, #abortionisnormal

Conclusions and Future Work

In this project, we were challenged to solve the problem of stance detection in tweets to be able to gain information about public opinion concerning the legalization of abortion topic. We concentrated on solving the problem of stance detection in general and found that the approach of extracting the sentiment of the tweets could help to increase the accuracy of detecting stances, so we focused on demonstrating that our approach of the two-phase architectures achieve better accuracy than previous state-of-the-art techniques and could be an effective tool to extract stances in tweets.

We searched for techniques that could improve different parts of the model and found that positional encoding helped to increase the performance of the attention layer, using pre-trained embeddings helped to boost the performance and the training, feeding the encoder sequentially the input and applying the attention to the previous state of the encoder as the query vector also increased the model's ability to detect stances from subjective sentences and gain more information about the relatedness of subjectivity and stance.

Incorporating the location of the twitter accounts as a feature alleviated the accuracy and performance of the model, which showed a correlation between that specific feature and mining political opinion.

We think that the general approach and lack of data had some limitations on extracting features that are specifically related to the topic such as gender, age, etc. which made it harder to do such integrations of data and analysis. We think in future work we can work on building and extracting metadata that is relevant to the topic and thus improve the analytic part of how the public opinion is divided around the topic, as we hope to enrich and enlarge our training data in future work we will be focusing on feature engineering, we did some text normalization but we hope to work on more complicated text normalization tasks like accent removal, substitution of contractions, acronym normalization, spell correction, this is very important sense we are dealing with open user inputs, etc.

we might as well try to incorporate sentence annotation information and graph learning models.

Thanks a lot !