Topic Classification of UN Speeches

BIA 667 - Project Report

Team Members:
Yash Jain
Abhinav Rakesh Kumar Shukla
Junaid Ali Sayyed

GitHub Repo

Introduction

This project aims to automatically classify UN speeches by topic using semi-supervised learning techniques. The classification of UN speeches by subject provides valuable insights into the priorities and concerns of different countries and how these priorities and concerns change over time. However, manually classifying thousands of speeches is a time-consuming and expensive task. This project will utilize a dataset of UN General Debate speeches from 1970 to 2016, publicly available on the Harvard Dataverse, containing over 1.5 million sentences. The dataset covers a wide range of topics, making it a useful resource for this project.

Literature Review

Past research has focused on various characteristics of UN speeches, including sentiment, style, and content. Alexander Baturo, et al (2018) conducted a study using natural language processing technologies to identify the drivers of the international development agenda over time. They found that key themes included economic growth, poverty reduction, social inclusion, and sustainability. The relative importance of these themes shifted over time, with a greater focus on sustainability and social inclusion in recent years.

Kohei Watanabe and Yuan Zhou (2013) demonstrated the effectiveness of semi-supervised learning techniques in classifying complicated text datasets, such as the UN speech collection. They provided the results of 2 techniques that they had implemented. The techniques are Newsmap and Seeded LDA the results of which are below:

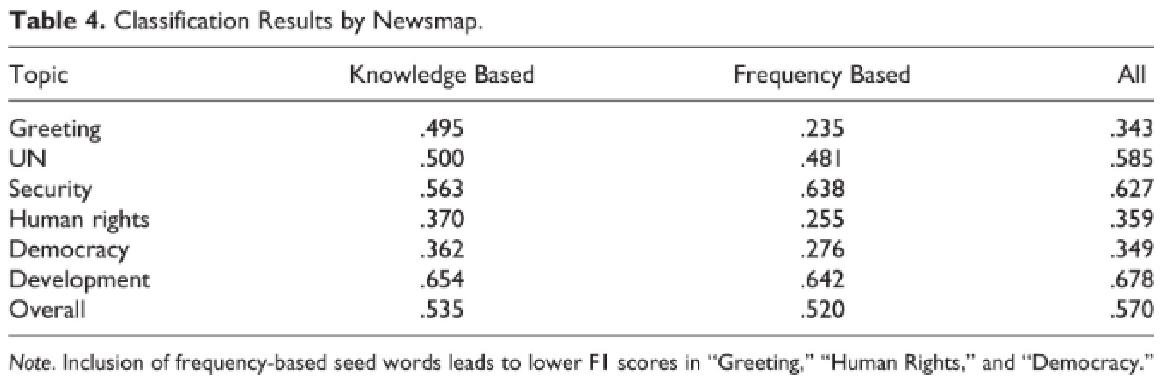


Table 4. Classification Results by Newsmap.

Topic	Knowledge Based	Frequency Based	All
Greeting	.495	.235	.343
UN	.500	.481	.585
Security	.563	.638	.627
Human rights	.370	.255	.359
Democracy	.362	.276	.349
Development	.654	.642	.678
Overall	.535	.520	.570

Note. Inclusion of frequency-based seed words leads to lower F1 scores in "Greeting," "Human Rights," and "Democracy."

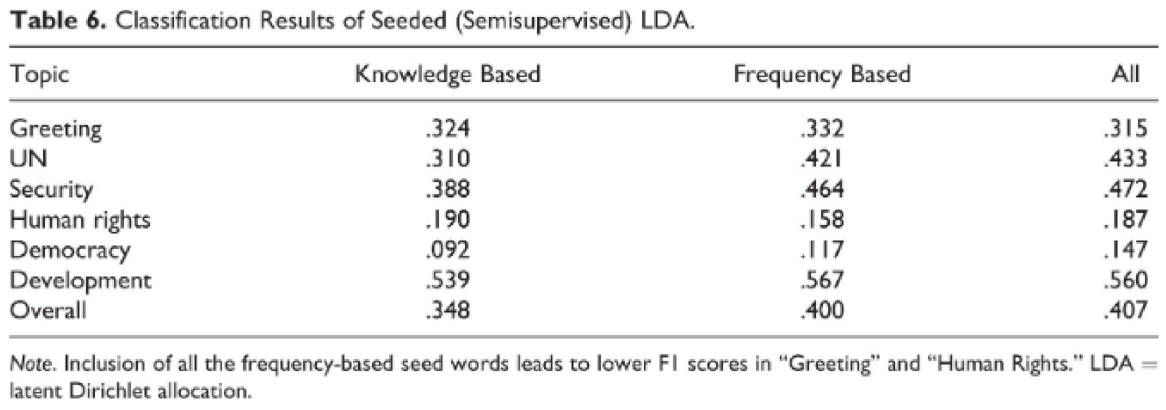


Table 6. Classification Results of Seeded (Semisupervised) LDA.

Topic	Knowledge Based	Frequency Based	All
Greeting	.324	.332	.315
UN	.310	.421	.433
Security	.388	.464	.472
Human rights	.190	.158	.187
Democracy	.092	.117	.147
Development	.539	.567	.560
Overall	.348	.400	.407

Note. Inclusion of all the frequency-based seed words leads to lower F1 scores in "Greeting" and "Human Rights." LDA = latent Dirichlet allocation.

Motivation

We have previously implemented multiple deep learning models in various domains like NLP and Computer Vision but have always restricted ourselves to the subarea of supervised learning. Having found this interesting dataset on UN General Debate speeches, we felt it was the perfect dataset to implement semi-supervised methods and expand our knowledge. This dataset has only a few existing publications and code available which made it more lucrative to us and became the motivation for this project.

Dataset

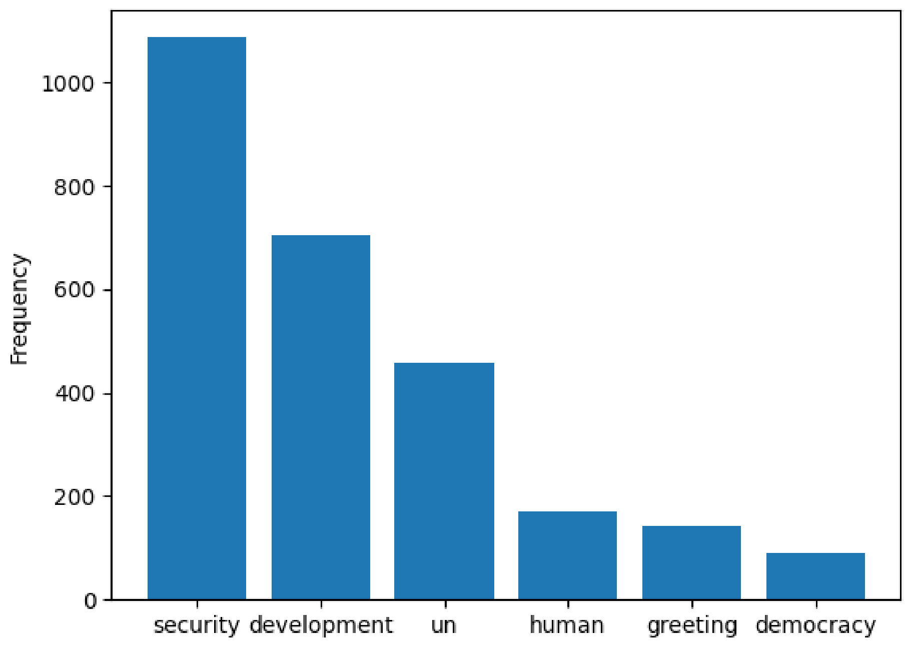
The dataset for this project is publicly available on the Harvard Dataverse and contains approximately 2 million sentences from UN General Debate speeches held from 1970 to 2016. The dataset consists of 1,86,337 unlabeled sentences and 2650 labeled sentences.

Our data contains 3 columns, the speech, the country that spoke that speech, and the year of the debate. The labeled data has 6 categories:

- Security
- Development
- UN
- Human
- Greeting

- Democracy

These labels are not equally represented which makes our data imbalanced. This is shown in the graph below:



Implementation

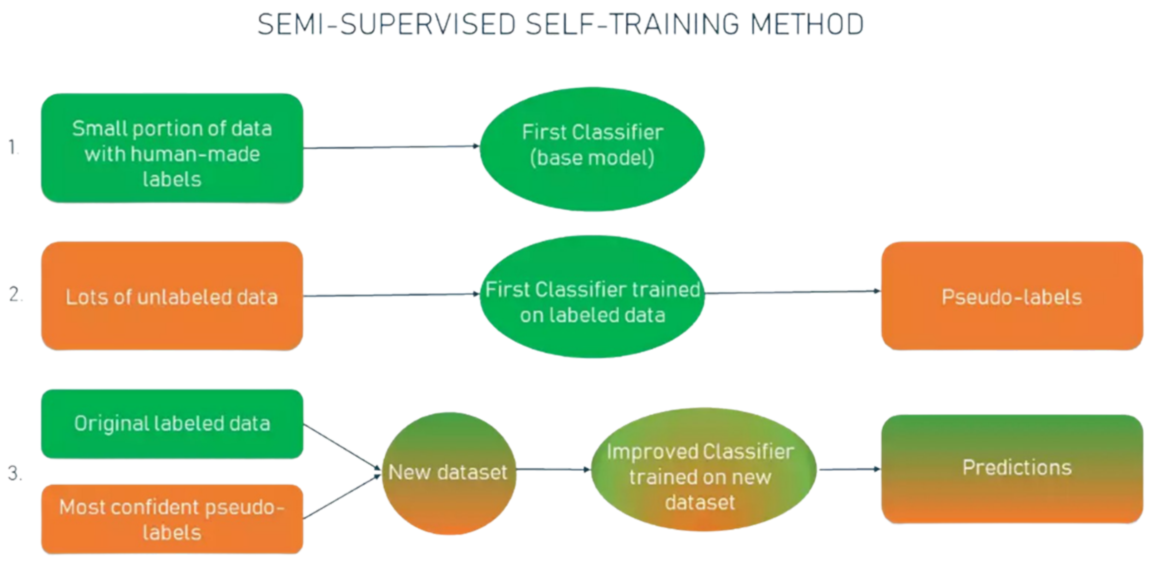
Pre-Processing

Before heading into the semi-supervised methodology or the models, we will first mention the pre-processing methods we have applied. Since our dataset contains speeches given by countries in UN General Debates, it is available to us in the form of text data. We apply the following methods:

- Remove punctuations
- Remove stopwords
- Remove numbers
- Remove whitespace
- Tokenize
- Remove sentences with lengths less than or equal to 2

These methods help us reduce noise in the data and improve the accuracy of the analysis. After applying all these pre-processing steps, we are left with a clean and structured dataset that we could use for further analysis.

Semi-Supervised Learning

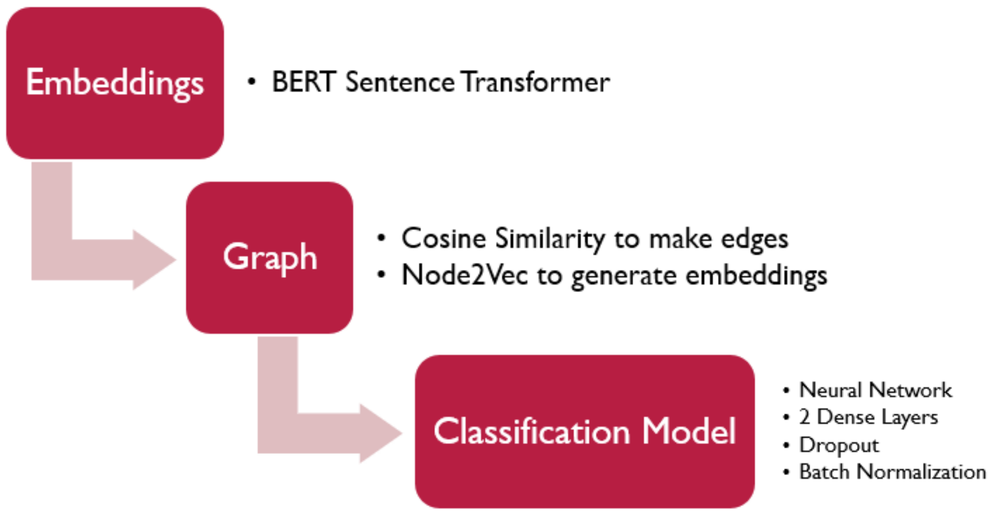

SEMI-SUPERVISED SELF-TRAINING METHOD
1
Small portion of data with human-made labels
First Classifier (base model)
2.
Lots of unlabeled data
First Classifier trained on labeled data
Pseudo-labels
3.
Original labeled data
Most confident pseudo-labels
New dataset
Improved Classifier trained on new dataset
Predictions

The above diagram best explains the method we have implemented. We first train a classifier on the existing labeled dataset. This gives us a base model. We use this base model to generate predictions of a portion of the unlabeled data. These predictions are also called pseudo-labels. We stack the pseudo labels and existing labeled data to generate a bigger dataset and train an improved classifier. This classifier is then used to further generate predictions and the process repeats till we get an acceptable result.

We have implemented this approach in 2 ways which we shall discuss next

Approach 1

Along with the semi-supervised methodology, we try to tackle this problem with graphs. The method is best illustrated with the following diagram:

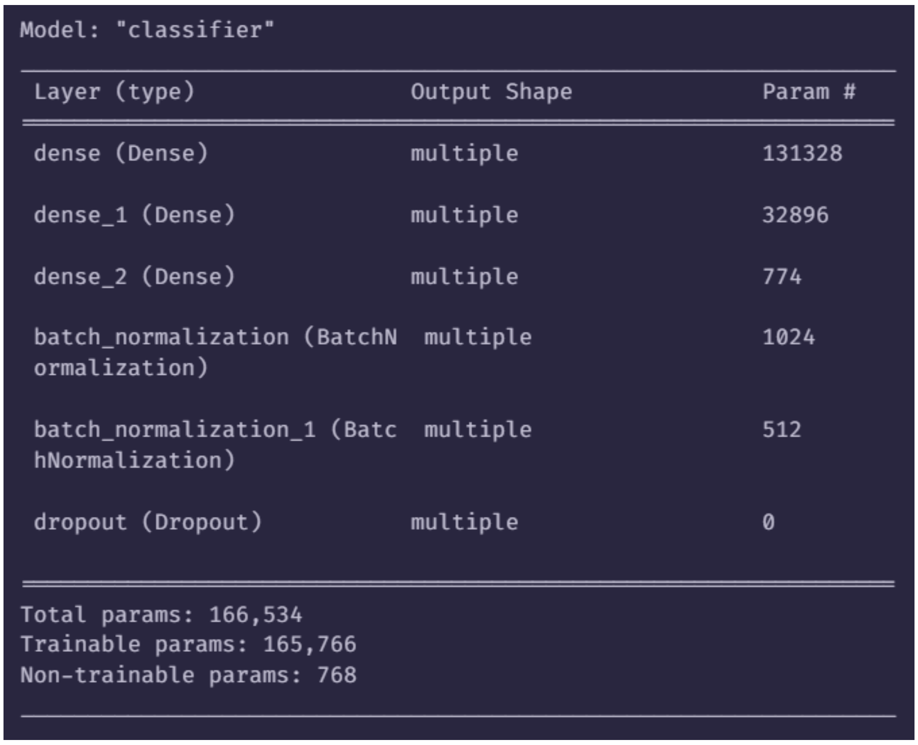


We convert our textual data into word embeddings using Bidirectional Encoder Representations from Transformers (BERT) Sentence Transformer provided by Hugging Face.

We then work on the formation of a graph. To form a graph we need 2 components: the nodes, and the edges. The nodes are each individual sentence (the word embeddings). We use cosine similarity as a criterion to form an edge between 2 nodes. We set a certain threshold above which an edge is made. We also used the features provided by the dataset to determine the edges but cosine similarity proved to be a better criterion than the given features. This is further explained later.

Post generating the graph, we need to get embeddings that we can use to classify the nodes into labels. We use Node2Vec that generates numeric embedding via 2nd order random walk. Node2Vec internally uses Word2Vec to generate node embeddings from the corpus of random walks it performs.

The node embeddings are then passed into a Classification model which is a multi-layer neural network of the following architecture:



```
Model: "classifier"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense (Dense)               multiple                  131328    
                                                                 
 dense_1 (Dense)             multiple                  32896     
                                                                 
 dense_2 (Dense)             multiple                  774       
                                                                 
 batch_normalization (BatchN  multiple                 1024      
 ormalization)                                                   
                                                                 
 batch_normalization_1 (Batc  multiple                 512       
 hNormalization)                                                 
                                                                 
 dropout (Dropout)           multiple                  0         
                                                                 
=================================================================
Total params: 166,534
Trainable params: 165,766
Non-trainable params: 768
_________________________________________________________________
```

To summarize, the model has 2 hidden layers with dropout and batch normalization applied after each layer. Using this model we generate pseudo labels for 2000 unlabeled sentences.

Below we mention the parameters we experimented with along with the values used:

Node2Vec Parameters:

- Embedding size: 64, 128, 256, 512, 1024

- Walk length: 10, 15, 20, 30, 80, 100
- Number of walks: 2, 10, 20, 80, 160
- Window size: 10, 20, 30
- Cosine similarity threshold: 0.9, 0.95, 0.97
- The criterion of edge formation: Cosine similarity, country, year

Classifier Parameter:

- Epochs: 50, 80, 100, 150, 200, 300
- Learning Rate: 0.9, 0.5, 0.1, 0.05, 0.01, 0.005, 0.001
- Dropout Rate: 0, 0.3, 0.5, 0.7
- Dimensions of the hidden layers: (256, 128), (512, 256, 128), (512, 256, 128, 64)
- Batch Normalization

Challenges Faced and Results of Approach 1

As mentioned earlier, we tried to use the features given in the dataset to form edges for the graph. We are provided with the country and the year of each sentence. While we did use these features, we observed that the loss was very high followed by an abysmal F1 score. This result is warranted by the observation that every country would most probably discuss every topic discussed by the UN General every year. Given this understanding, it is easily understandable why the model did not perform well. The model could not find any relation between the topics and the nodes since the edges were in no manner relevant to the required result. The result of this experiment is as follows:

Edge Criterion	**Train Loss**	**Train F1**	**Eval Loss**	**Eval F1**
Cosine Similarity	1.4362	0.1190	1.5119	0.1156
Country	2.1264	0.0953	2.6452	0.0912
Year	2.0142	0.0942	2.4876	0.0874
Country + Year	3.1002	0.0412	2.9931	0.0349

This leaves us with just cosine similarity as the criterion. The high loss and low F1 score made us think if a different criterion would suit the model better. The lack of a better criterion is one of the challenges we face.

We experimented with a lot of parameters with a variety of values. We noticed that none of the parameters had any effect on the scores. This is demonstrated below:

Learning Rate	**Train Loss**	**Train F1**	**Eval Loss**	**Eval F1**
0.001	1.4475	0.1230	1.4795	0.1226

0.1	1.4244	0.1227	1.5099	0.1156

Dropout Rate	**Train Loss**	**Train F1**	**Eval Loss**	**Eval F1**
0.3	1.4362	0.1190	1.5119	0.1156
0	1.4368	0.1236	1.5144	0.1156

Node Embedding Size	**Train Loss**	**Train F1**	**Eval Loss**	**Eval F1**
1024	1.4500	0.1223	1.4765	0.1295
64	1.4475	0.1230	1.4795	0.1226

Walk Length	**Train Loss**	**Train F1**	**Eval Loss**	**Eval F1**
100	1.4352	0.1232	1.4892	0.1156
10	1.4545	0.1239	1.5369	0.1172

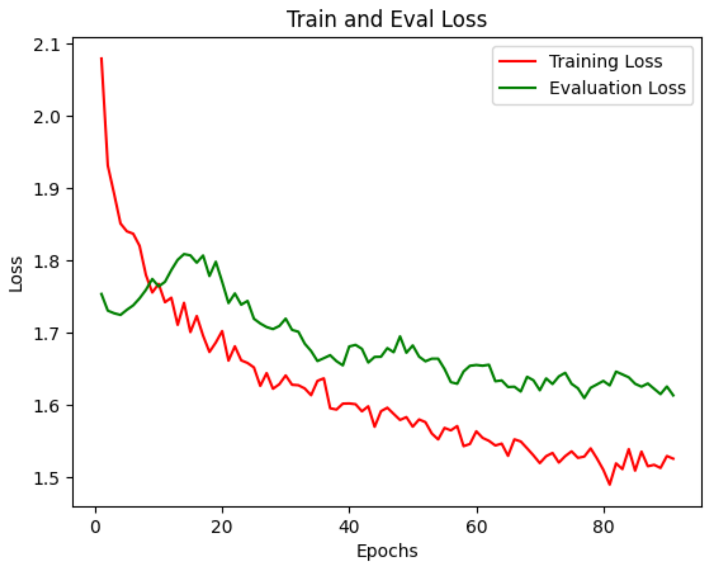
Number of Walks	**Train Loss**	**Train F1**	**Eval Loss**	**Eval F1**
160	1.4624	0.1245	1.4566	0.1072
2	1.4703	0.0964	1.4534	0.1003

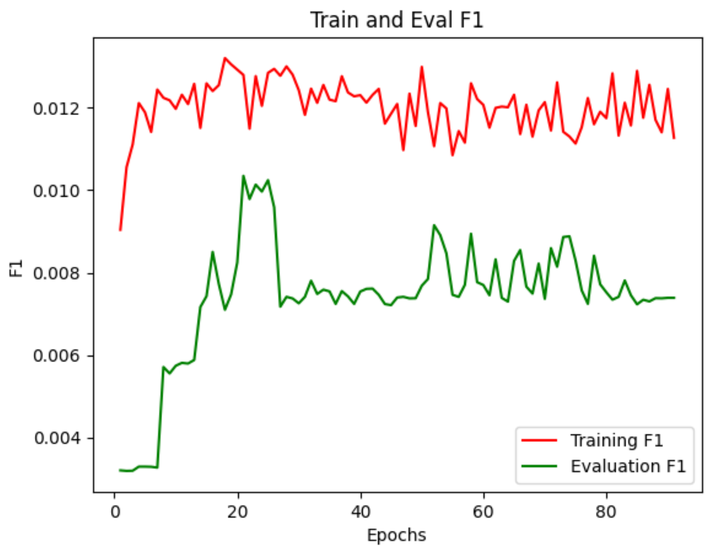
Window Size	**Train Loss**	**Train F1**	**Eval Loss**	**Eval F1**
30	1.4624	0.1245	1.4566	0.1072
10	1.4545	0.1239	1.5369	0.1172

We also tried to change the threshold on cosine similarity but there were no pairs formed on a threshold of 0.97 and only 3 pairs formed on a threshold of 0.95. Hence we stuck with a threshold of 0.9.

During our experimentation, we observed that changing the parameters such as the cosine similarity threshold, embedding size, walk length, number of walks, and window size did not significantly impact the F1 score and loss. This was surprising as we expected that changing these parameters would have a significant impact on the performance of the model. However, this lack of sensitivity to the parameters might indicate that the underlying graph structure is not strong enough to produce significant changes.

Here are the plots for the loss and F1 score of both training and validation data:



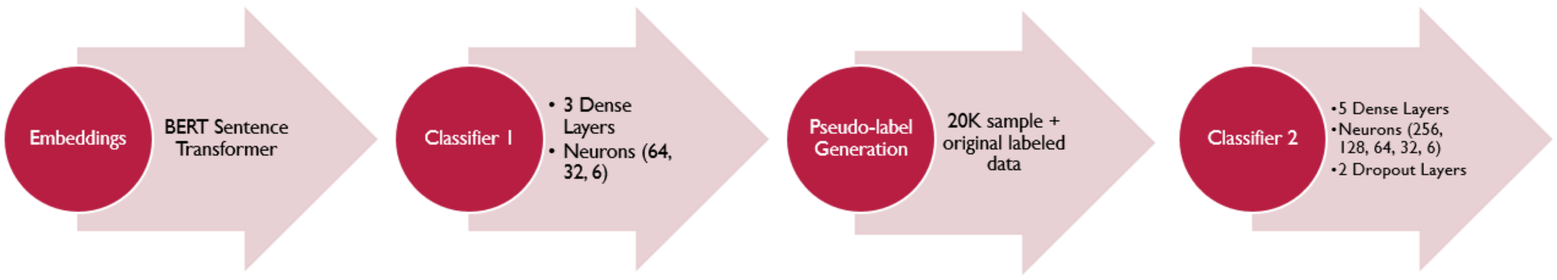


As we can see, the F1 score does not improve beyond a certain point and even though the loss is decreasing, we can observe overfitting occurring despite the usage of dropout and batch normalization.

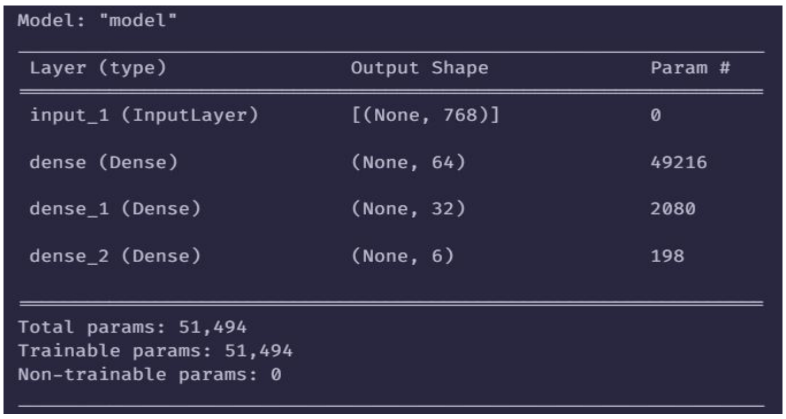
Since the model produced a very low F1 score despite all our experiments, which in terms gives a very low confidence in the correctness of the labels generated, we decided to not pursue this approach further and did not generate the pseudo-labels.

Approach 2

The second approach also includes generating text embeddings using BERT. The flowchart of the approach is mentioned in the below figure.



For the generation of embeddings, the SentenceTransformer BERT Model is being used, a Python framework for state-of-the-art sentence, text, and image embeddings. The embeddings are necessary to understand the context of the sentences and make it easier to process the input as vocabulary would be large in our case. The first step was to generate text embeddings on the entire labeled dataset (around 2000 samples). The embeddings were passed to a base neural network comprising 3 Dense layers. The summary of the model is shown in the below figure.

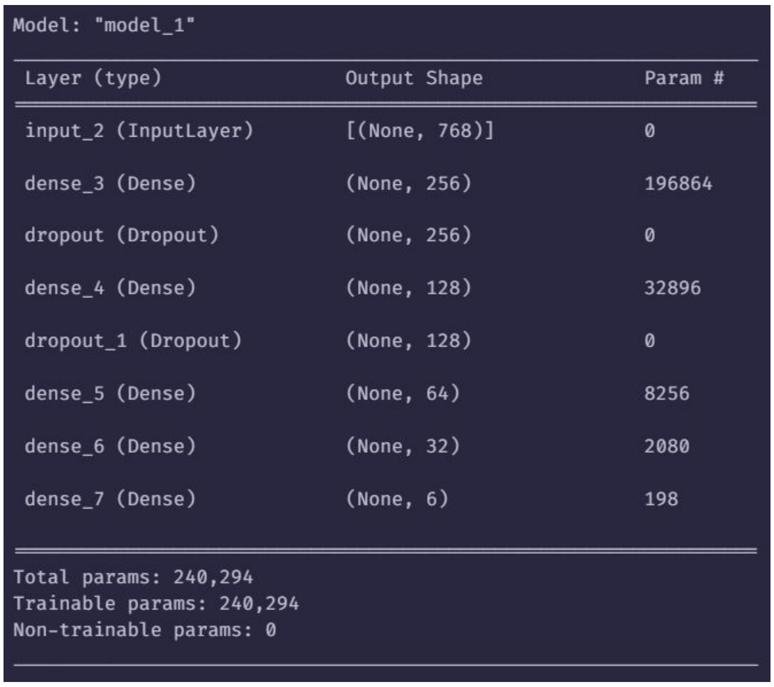


```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 768)]             0

 dense (Dense)               (None, 64)                49216

 dense_1 (Dense)             (None, 32)                2080

 dense_2 (Dense)             (None, 6)                 198

=================================================================
Total params: 51,494
Trainable params: 51,494
Non-trainable params: 0
_________________________________________________________________
```

The above classifier ran for 500 epochs with the given configuration. It was observed that the model ended up with the following results:

Train Accuracy	Train F1	Validation Accuracy	Validation F1
0.9993	0.9993	0.6316	0.4820

After the training, 20,000 samples from unlabelled data were labeled using the trained classifier. The labeling took around 2 hours to execute on Colab. Labeled along with newly labeled samples were stacked to make a significant amount of data for training. In the next steps, a more complex neural network was built, comprising 5 Dense layers as shown in the below figure. Overfitting was observed, and to tackle that two dropout layers were integrated within the model architecture.



```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 768)]             0

 dense_3 (Dense)             (None, 256)               196864

 dropout (Dropout)           (None, 256)               0

 dense_4 (Dense)             (None, 128)               32896

 dropout_1 (Dropout)         (None, 128)               0

 dense_5 (Dense)             (None, 64)                8256

 dense_6 (Dense)             (None, 32)                2080

 dense_7 (Dense)             (None, 6)                 198

=================================================================
Total params: 240,294
Trainable params: 240,294
Non-trainable params: 0
_________________________________________________________________
```

Below we mention the parameters we experimented with along with the values used:

Hyper-Parameters	**Experimented Values**	**Best Parameter(s)**
Learning Rate	0.00001, 0.0001, 0.001, 0.1	0.00001
Optimizer	Adam, Adagrad, RMSprop	Adam
Epochs	50, 100, 250, 300, 500	100
Batch Size	16, 32, 64	32
Dropout Rate	0.1, 0.3, 0.2, 0.5	0.3, 0.2 (for 2 layers)

Challenges Faced In Approach 2

Imbalanced Data

Imbalanced data is a common issue in machine learning, and it affects the performance of the models. During analysis, we found that our dataset also suffered from this problem, with a significant imbalance in the number of samples in different classes. To tackle this, we tried various techniques like oversampling using Synthetic Minority Over-sampling Technique (SMOTE). This oversampled the minority classes samples such as democracy, human, and greeting in our case.

Embedding Model

The choice of embedding model is a critical aspect of any NLP task. As mentioned earlier we needed to generate embeddings and we had to choose between BERT, Word2Vec, and GloVe. Since BERT provides contextual information as compared to other options, we had to take the whole sentence as input. After thorough analysis, we found that the BERT Sentence Transformer provided the best results for our task.

Lack of Resources

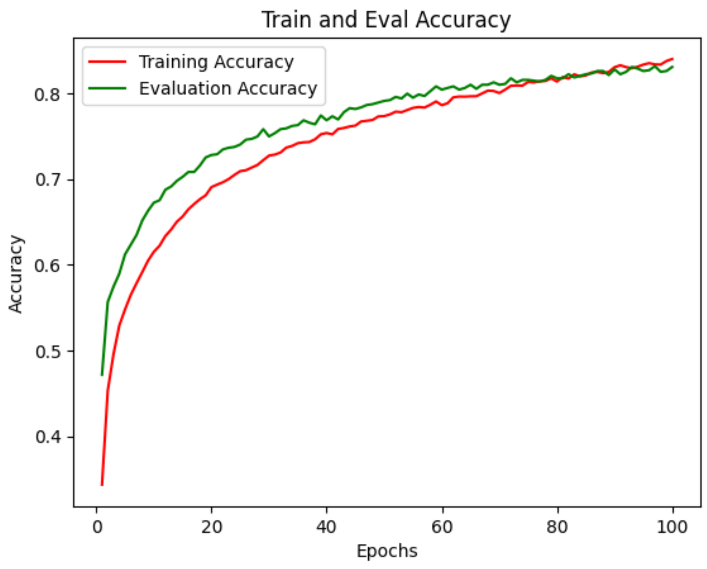
The lack of resources was another hurdle we faced during our experiments. With limited RAM and processing power on Google Colab and our own systems, we had to perform batch training. We had 12 GB of available RAM on Google Colab with 78 GB of Disk Space and 64 GB of available RAM on Saturn Cloud with only 3GB of disk space. Therefore, on each step, we kept saving the embeddings in batches rather than encoding the entire dataset.

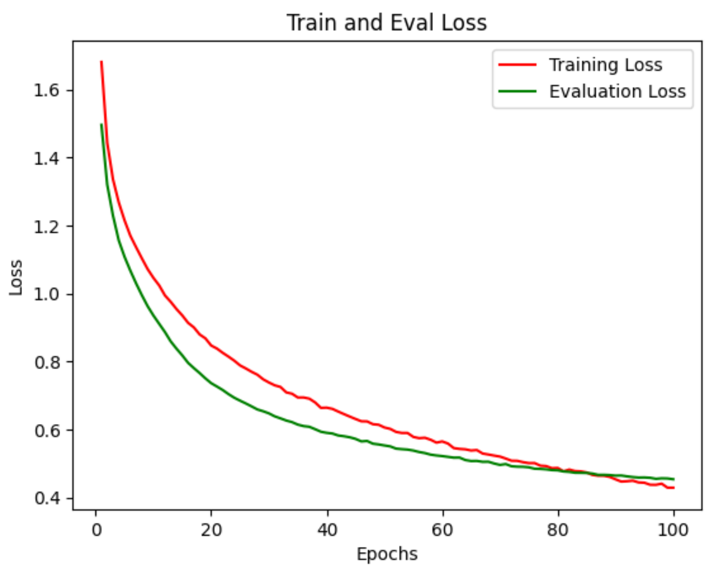
Results of Approach 2

Since data was imbalanced, we decided to go with F1-score as the metric for evaluation, as it combines precision and recall into a single metric. We tried to achieve as high an F1 score as possible. Post-completion of the training, the results of the final classifier are listed below.

Train Loss	**Train Accuracy**	**Train F1 score**	**Valid Loss**	**Valid Accuracy**	**Valid F1 score**
0.4293	0.8398	0.8005	0.4542	0.8305	0.7826

Following are the plots for Accuracy and Loss for both Training and Validation:





Overall, Approach 2 seemed to be a promising solution for our classification problem. The semi-supervised training method allowed for the use of a large amount of unlabeled data to improve the performance of the model, while the evaluation metrics showed that the model was performing well.

Conclusion

Having implemented 2 different models to approach semi-supervised learning, we now have a better understanding of the problem. While approach 1 did not provide us with good results, we received good scores from approach 2. Comparing Kohei Watanabe and Yuan Zhou (2013)'s results, our model performs significantly better as shown below:

	Our F1 Score	**Kohei et al Newsmap F1 Score**	**Kohei et al Seeded LDA F1 Score**
Greeting	0.7716	0.343	0.315
UN	0.8027	0.585	0.433
Security	0.8772	0.627	0.472
Human rights	0.7701	0.359	0.187
Democracy	0.6451	0.349	0.147
Development	0.8283	0.678	0.560
Overall	**0.7826**	**0.570**	**0.407**

Future Scope

We have a few options that might improve the model's performance:

1. We can use more labeled data to train a better base model. This data can be fetched from Kaggle and other dataset resources publicly available.

2. We can find more criteria to connect nodes in the graph implemented in Approach 1. There might be some criterion that is more efficient or more sensible for this use case

3. For now, since we have received a sufficient F1 score, we have taken all the pseudo labels generated by the model. But we can use some metrics to judge the confidence of the generated labels. One such metric could be Shannon Entropy. Using only the confident labels might improve the subsequent models.

References

[1] What Drives the International Development Agenda? An NLP Analysis of the United Nations General Debate 1970-2016 Alexander Baturo, Niheer Dasandi, & Slava J. Mikhaylov. (2018)

[2] Kohei Watanabe and Yuan Zhou (2013). Theory-driven analysis of large corpora: Semi supervised topic classification of the UN speeches.

[3] “Semi-Supervised Learning, Explained with Examples,” AltexSoft, Mar. 18, 2022. https://www.altexsoft.com/blog/semi-supervised-learning/