

**Cyberbully Detection and Analytics On Online Media**

**INTERNSHIP REPORT**

**Quarter IV (Year 1)**

***Submitted by***

**SHREESHAA S E0221026**

***In partial fulfillment for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

**in**

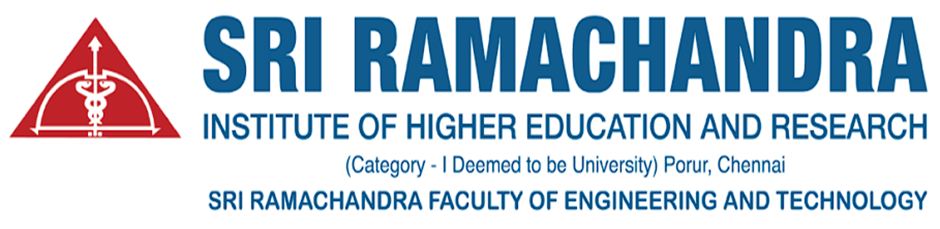
**COMPUTER SCIENCE AND ENGINEERING**

**(Cyber Security & Internet of Things)**

**Sri Ramachandra Engineering and Technology**

**Sri Ramachandra Institute of Higher Education and Research, Porur, Chennai -600116**

**JULY, 2020**



**BONAFIDE CERTIFICATE**

Certified that this project report **“Cyberbully Detection and Analytics On Online Media”** is the bonafide work of **SHREESHAA S Reg No. E0221026** who carried out the internship work under my supervision.

**Signature of Faculty Mentor Signature of Vice-Principal**

|  |  |
| --- | --- |
| **Dr. Jayanthi Ganapathy**  Assistant Professor  Sri Ramachandra Engineering and Technology  Porur  Chennai-600116 | **Prof. M. Prema**  Vice-Principal  Sri Ramachandra Engineering and Technology  Porur  Chennai-600116 |

**Evaluation Date:**

**Title** **Page**

1. Domain Introduction 5
   1. Cyberbullying 5
   2. Literature Review and Analysis 5
   3. Harmful effects 6
2. Objective 8
3. Work Flow and Architecture 9
   1. Data Mining 9
      1. Wikipedia Corpus 9
      2. Book Corpus 11
      3. Building the Data-set 13
   2. Building the Model 15
      1. Data preparation 15
      2. Implementation 18
      3. Classifier 22

3.3 API Integration 23

1. Tools and Technology Used 24
   1. Git and GitHub 24
   2. PyTorch 24
   3. Amazon EC2 24
   4. Amazon Lake Formation 25
2. Results and Output 26
   1. Data analysis on Cyberbullying 26
   2. Model training results 27
   3. API Integration Example with Discord.js 28
3. Timeline 30
4. Scope for further enhancement 31

Conclusion 32

References 33

**ACKNOWLEDGEMENT**

I express my sincere gratitude to our Chancellor, Vice-Chancellor and our sincere gratitude to our Provost **Dr. Raju** and our Vice-Principal **Prof. Prema** for their support and for providing the required facilities for carrying out this study.

I wish to thank my faculty mentor, **Prof. Jayanthi Ganapathy** Department of Computer Science and Engineering, Sri Ramachandra Engineering and Technology for extending help and encouragement throughout the project. Without his/her continuous guidance and persistent help, this project would not have been a success for me.

I am grateful to Department of Computer Science and Engineering, Sri Ramachandra Engineering and Technology, our beloved parents and friends for extending the support, who helped us to overcome obstacles in the study.

**Images**

1. Figure 1 - Survey on Cyberbullying
2. Figure 2 - Data Mining Flow
3. Figure 3 - Wiki HTML Files
4. Figure 4 - Converted Text Files
5. Figure 5 - JSON Data from Google API
6. Figure 6 - Book Crawler Running
7. Figure 7 - Books Converted To Text Files
8. Figure 8 - S3 Data Lake
9. Figure 9 - Data Stored in Lake
10. Figure 10 - Loading the Custom Data-set
11. Figure 11 - 10,000 vocabulary built using WordPiece
12. Figure 12 - Preprocessed Data
13. Figure 13 - Attention Architecture
14. Figure 14 - Transformer Model
15. Figure 15 - BERT Training
16. Figure 16 - BERT Classifier Training
17. Figure 17 - Most Used Hate words
18. Figure 18 - Forms of Cyberbullying
19. Figure 19 - Classifier Accuracy
20. Figure 20 - Classifier Loss
21. Figure 21 - Language Model Loss
22. Figure 22 - Deleting messages and warning users
23. Figure 23 - Logging Cyberbullying related events
24. Figure 24 - Timeline June 29 - July 16
25. Figure 25 - Timeline July 10 - July 29
26. **Domain Introduction**

**1.1 Cyberbullying**

In simple words, cyberbullying is bullying with the use of digital technologies. It takes place often on digital platforms like social media, messaging platforms, gaming platforms and multiplayer games. It involves offensive behaviour intended to shame, scare or anger a person. For example :

* Sending hate speech, hurtful, abusive or threatening messages, images, videos and other media on social platforms.
* Impersonating someone and sending mean messages to others on their behalf or through fake accounts.
* Spreading lies about or posting embarrassing photos or videos of someone on social media.

Unlike face-to-face bullying, cyberbullying leaves a digital footprint and can be used as evidence to help combat the issue.

**1.2 Literature Review and Analysis**

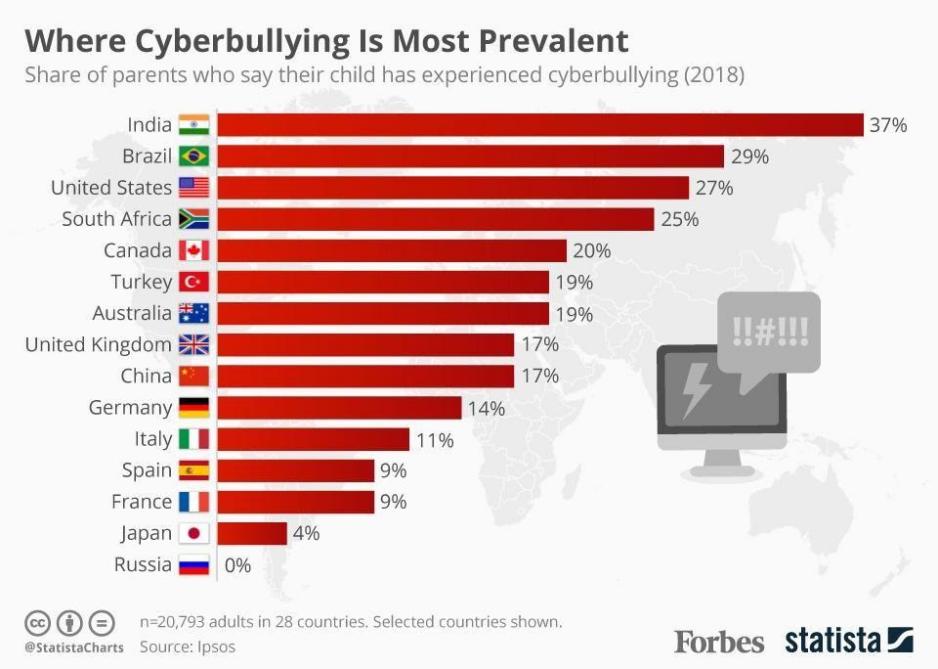


Figure 1 - Survey on Cyberbullying

As shown in Fig. 1, a survey was conducted on 20,000 parents worldwide about high-risk online platforms, “65% single out cyberbullying on social media as their biggest fear. Other common threats include text messaging (38%) and chat rooms (34%).” Clearly India seems to be most affected by cyberbullying, with factors such as high population and easy access to internet in recent times.

**1.3 Harmful effects**

It’s no surprise that cyberbullying, like regular face-to-face bullying can be detrimental to the mental health of a person. Experiences of cyberbullying can cause someone to feel ashamed, nervous and insecure about the opinions of others. It can lead to withdrawal from social interactions and make them feel subdued.

It could destroy self-confidence and motivation to do the things are usually enjoyable and create feelings of isolation from the people you love and trust. For young people, skipping school is another common effect of cyberbullying and they may turn to substances like alcohol and drugs or violent behaviour to deal their psychological and physical pain. The effects of cyberbullying on mental health can vary depending on the medium through which it happens.

1. **Objective**
2. Data Mining and Preparation
3. Model Training and Optimization
4. Model inference through a HTTP API

Making use of the digital evidence which cyberbullying leaves behind, we can combat this issue perhaps at the very early stages. With the tech and tools we have today, it’s possible to make use of machine learning and artificial intelligence to detect hate speech and offensive content in a text message.

Hate speech detection through machine learning can be useful for content moderation in different social media platforms. The objective is to build a machine learning model which takes text as parameters and classifies the content as offensive, hateful or normal through a scale of 0 to 1. This classification can be useful to implement auto moderation in social platforms such as warning the users or blocking the messages from being sent.

To provide a single solution to all platforms, an API such a simple HTTP API can be could be used to communicate with language model for receiving model classification on the input text. An HTTP is easy to integrate in social and gaming platform to check cyberbullying.

**3. Work Flow and Architecture**

**3.1 Data Mining**

To train the language model, a data-set containing text should be created through mining it from the required sources. This could be accomplished in 2 different ways :

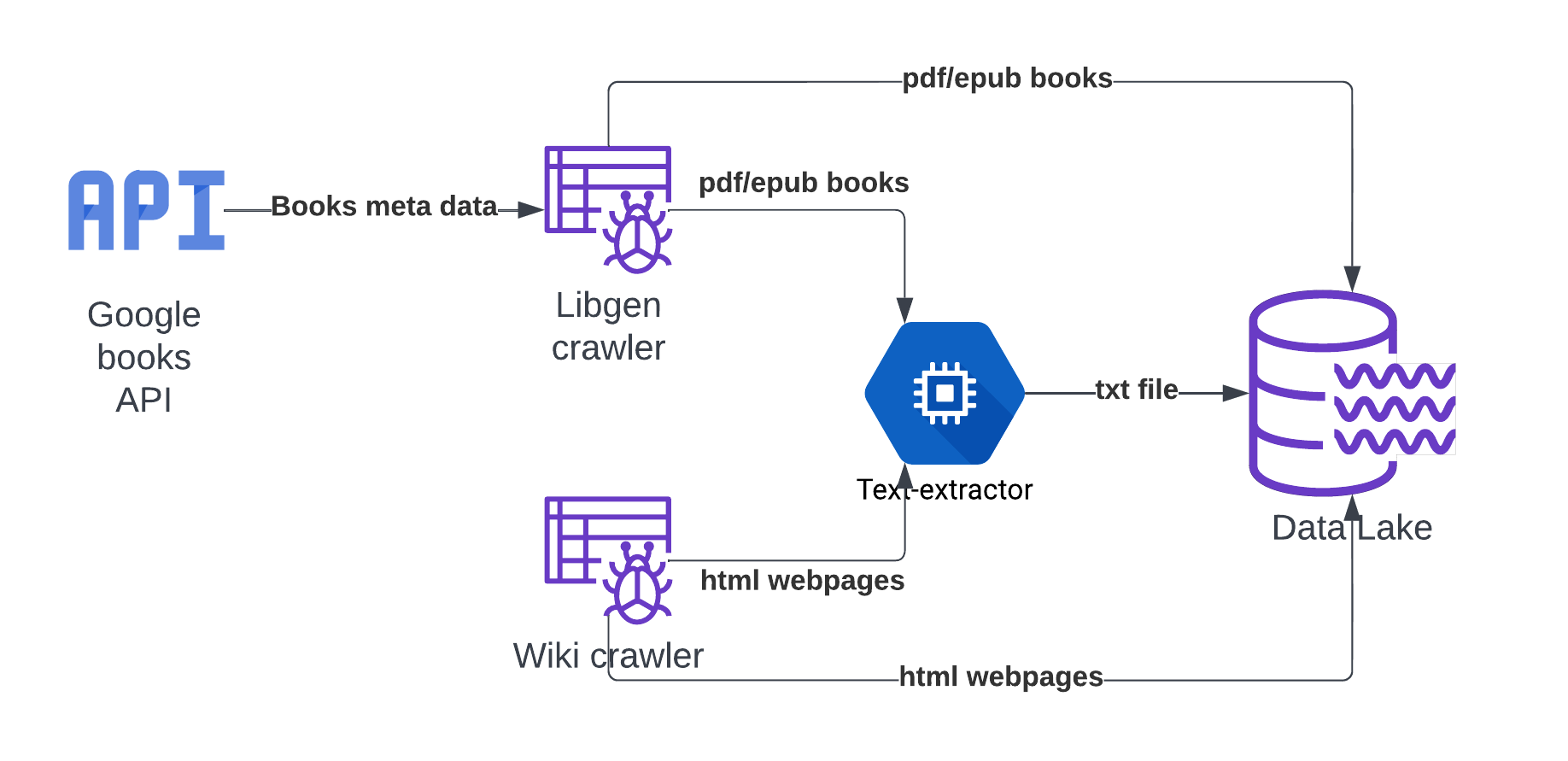


Figure 2 - Data Mining Flow

* + 1. **Wikipedia Corpus**

Wikipedia is a non-profit organization which provides articles which contain reference links and hosts other projects. It is a reliable source for data mining, hence a web crawler was built in python to crawl through Wikipedia. The web-pages are queried using the requests module and the HTML content are scraped using the BeautifulSoup module as shown in Fig. 3 and continues crawling and collecting text data in Wikipedia. The collected text for each web-page article is saved to their respective .txt files.

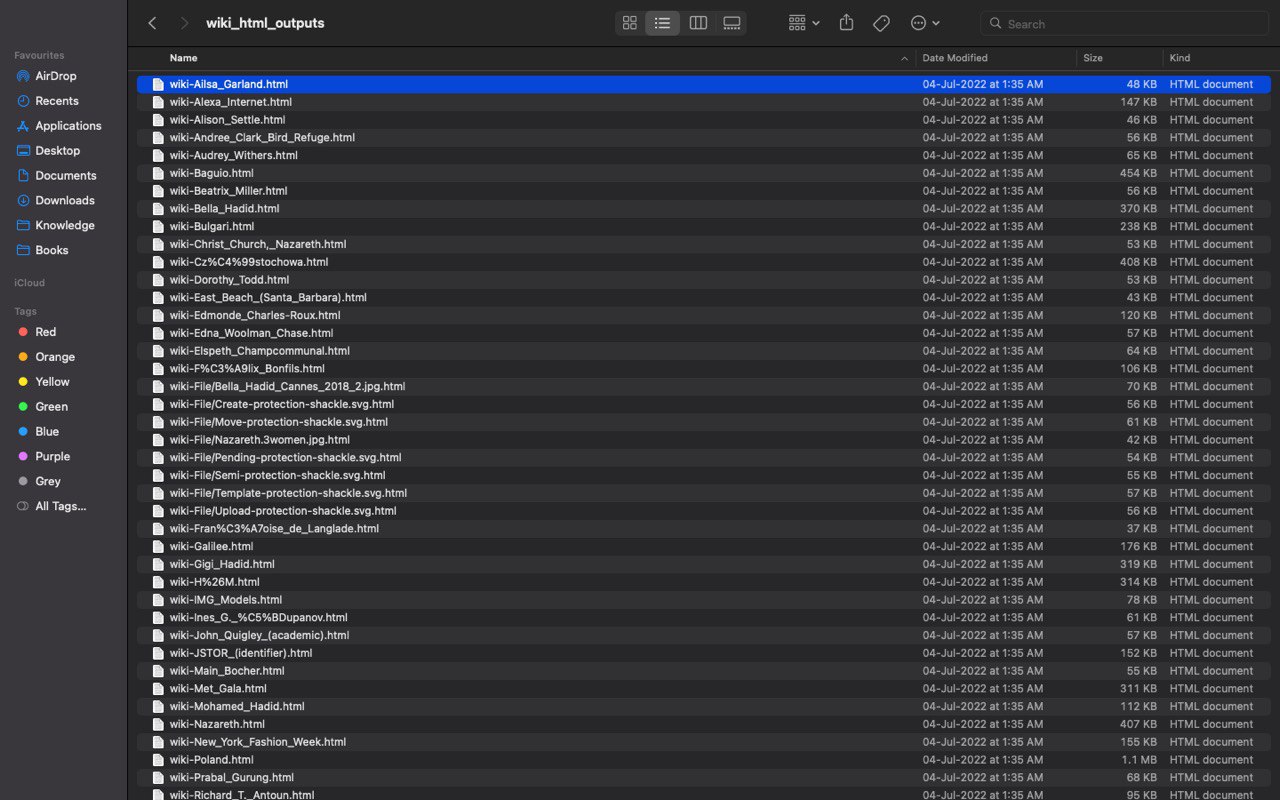
****

Figure 3 - Wiki HTML Files

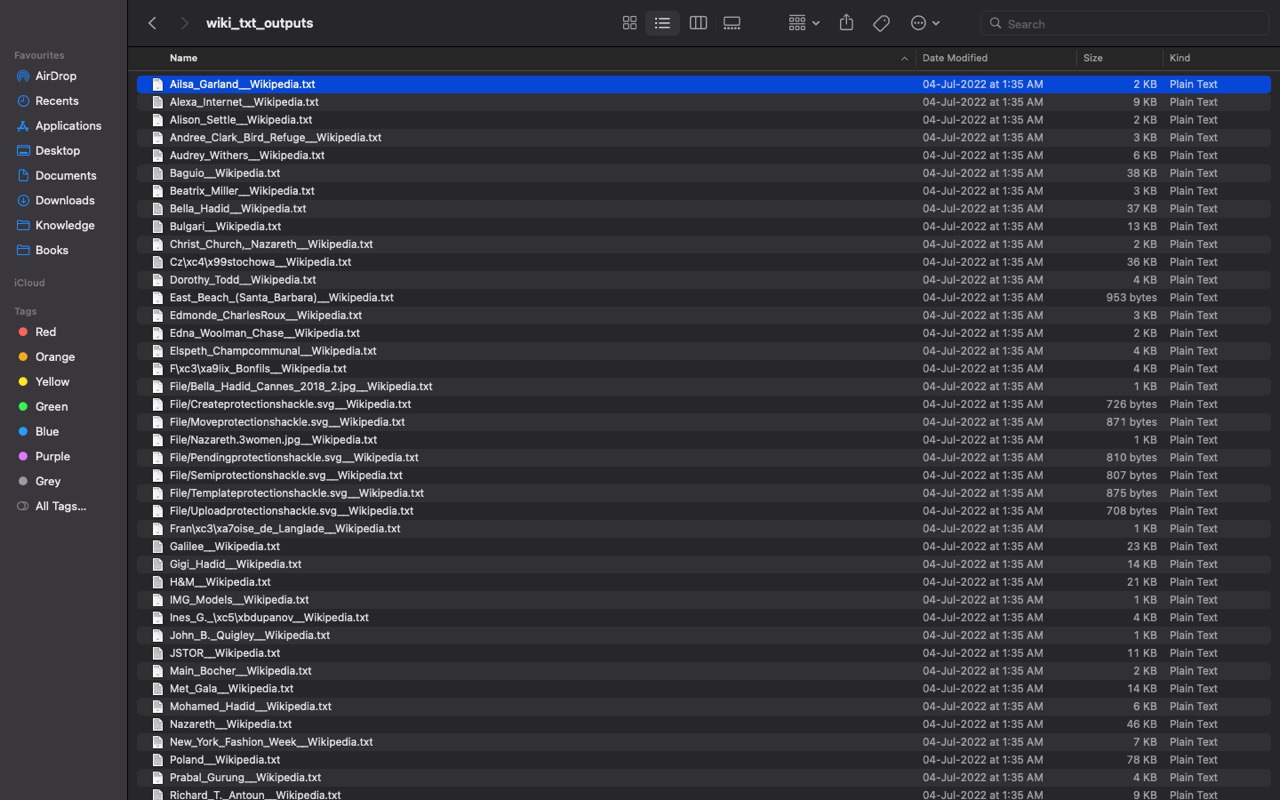


Figure 4 - Converted Text Files

This way, a crawler was used to mine text data from Wikipedia, and proves to be a reliable source for building a data-sets.

**3.1.2 Book Corpus**

From the perspective of data mining, digital books are huge chunks of text which are an excellent source of data. This proves to be a reliable source for building the data-set as well.

However the method to retrieve text data from books is not very straight forward. It requires querying a list of books from an API such as Google Books API. Google Books API provides a feature to search books by various metadata such as name, genre, author and so on. For the purpose of mining data, 5 genres were selected, and a book list was constructed by querying the API. The collected metadata about books were used to retrieve the digital books through a python crawler.

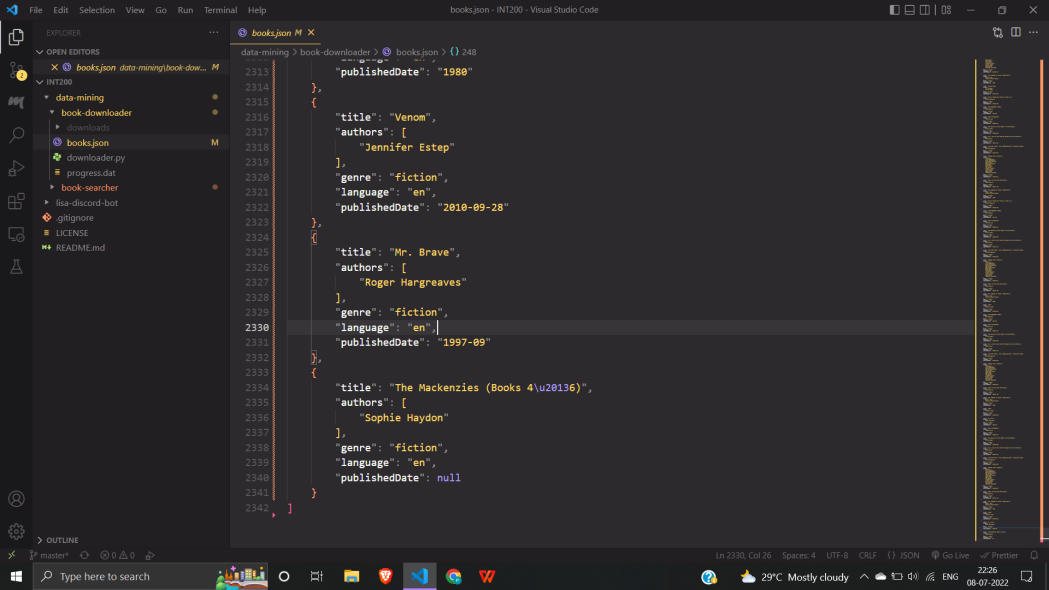


Figure 5 - JSON Data from Google API

Library Genesis is an online book library which provides books in various formats. Using the metadata that was collected, a python crawler was used to search the website for each book. The crawler was built to collect download links of .epub and .pdf formats were collected and downloaded, as shown in Fig. 6.

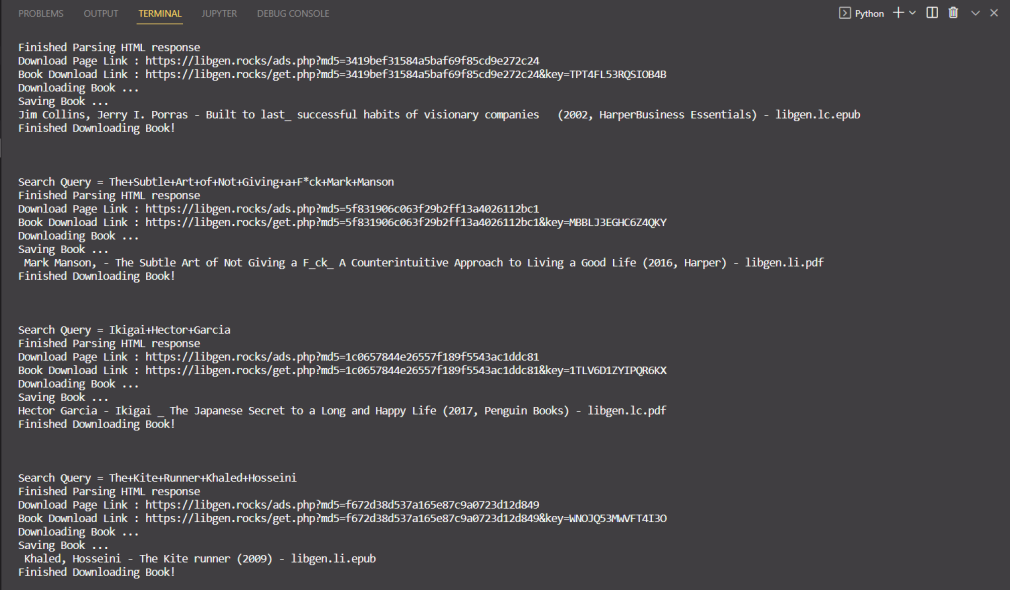


Figure 6 - Book Crawler Running

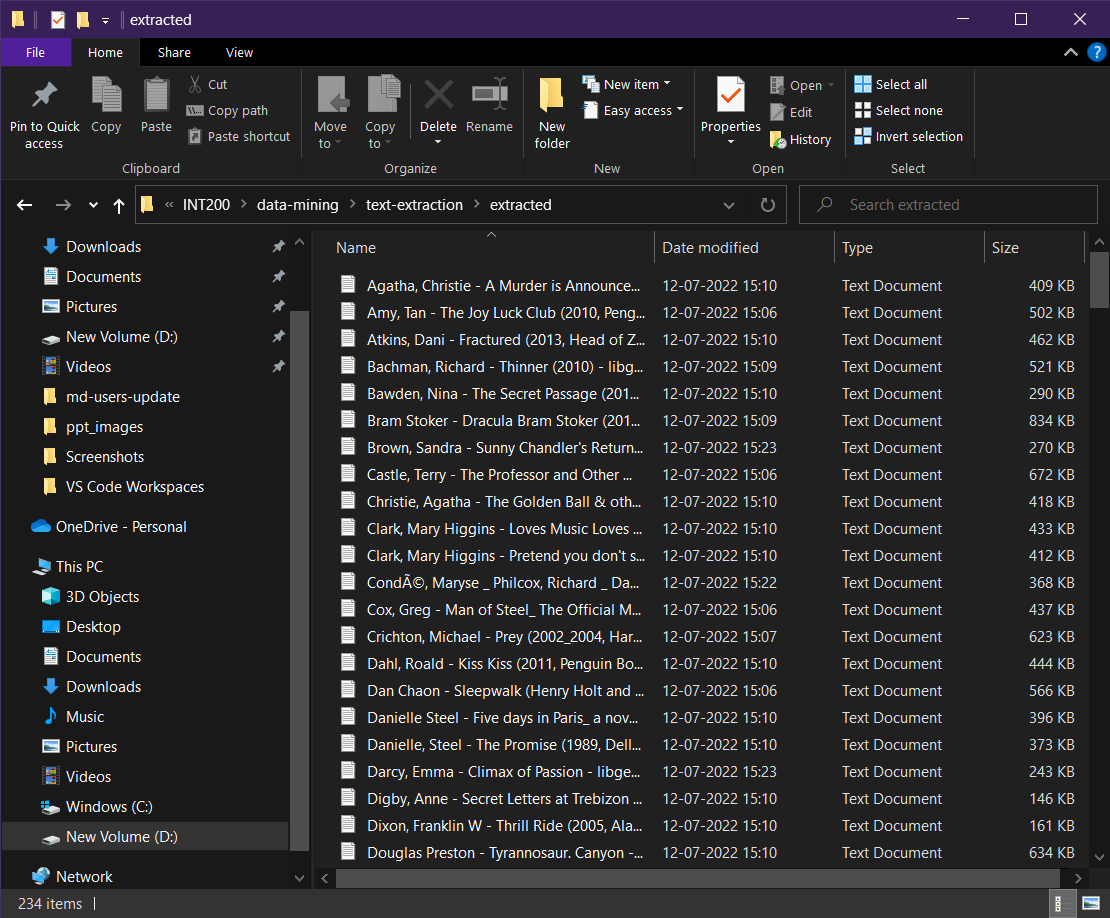


Figure 7 - Books Converted To Text Files

One last step remains before the downloads books can be useful for building the data-set. The text has to be extracted from each book into .txt files just like for the wiki articles. ebooklib is useful to extract text content from .epub books. Similarly, the PyPDF2 python module can be used to extract from a .pdf file.

**3.1.3 Building the Data-set**

Since its not practical to download and save the text data locally, it’s a better approach to directly upload the content to AWS Data-Lake for storage purposes. The Data-Lake will be the data-set for training the Language Model.

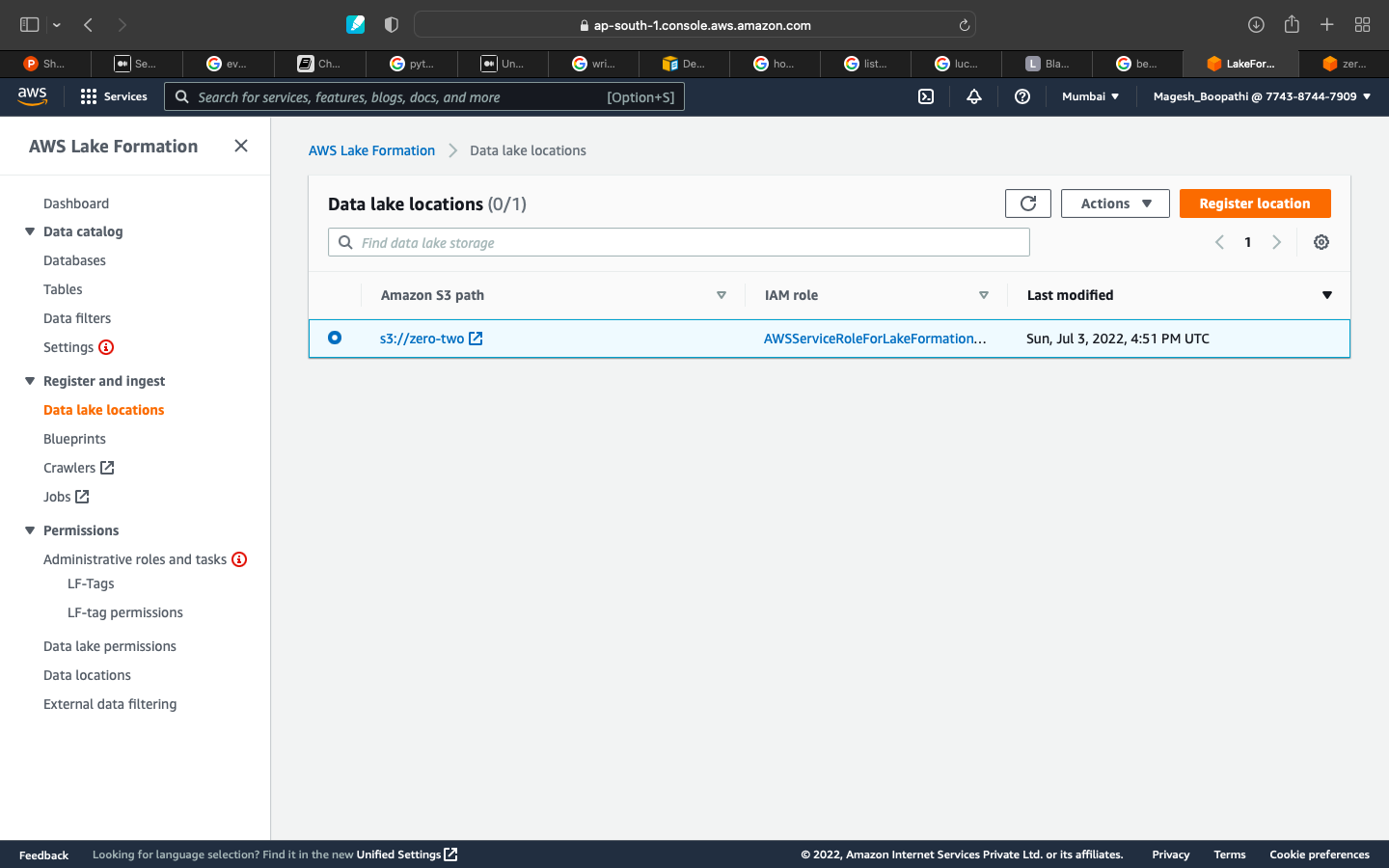
****

Figure 8 - S3 Data Lake

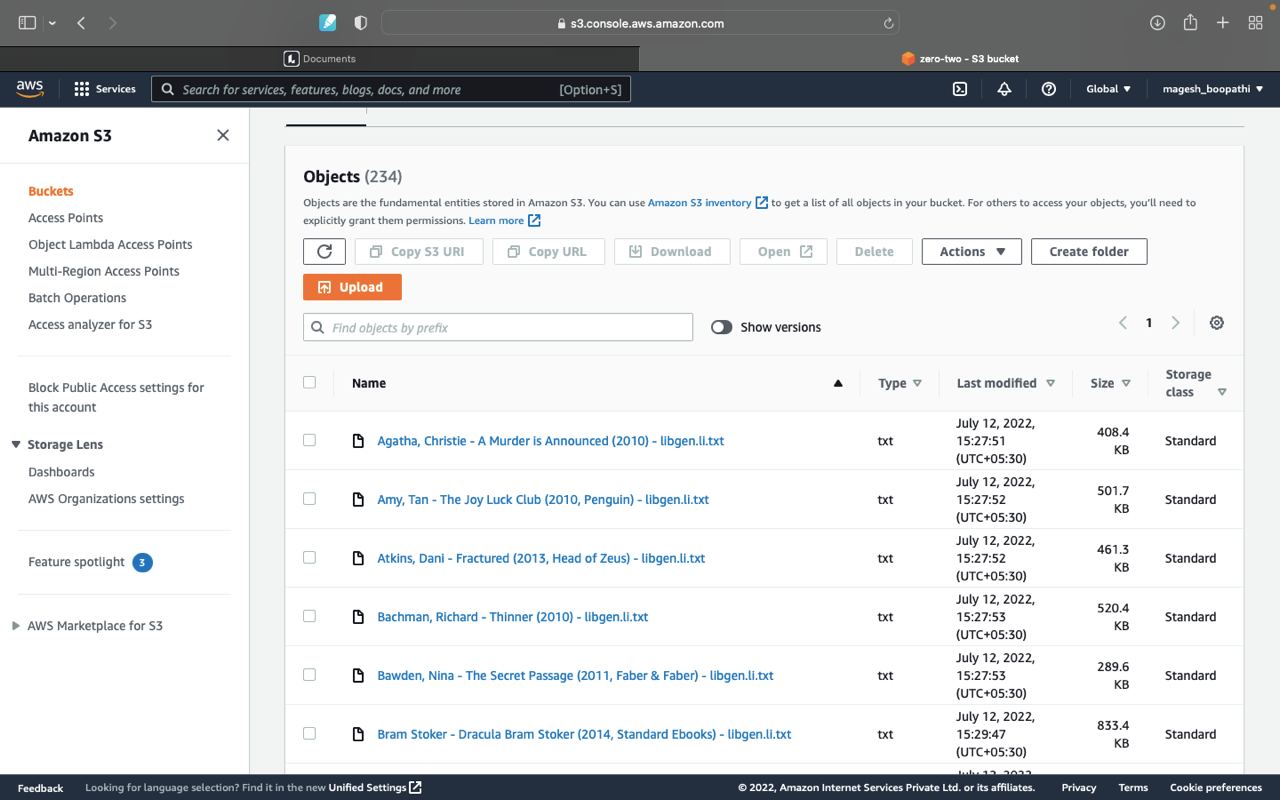
****

Figure 9 - Data Stored in Lake

**3.3 Building the model**

**3.3.1 Data preparation**

In order to pre-train any language model, a large amount of words from a particular language is required. BERT was built using Books Corpus (800M words) and English Wikipedia (2500M words). [1]

In order to obtain similar result, we built our own book corpus and English Wikipedia corpus.

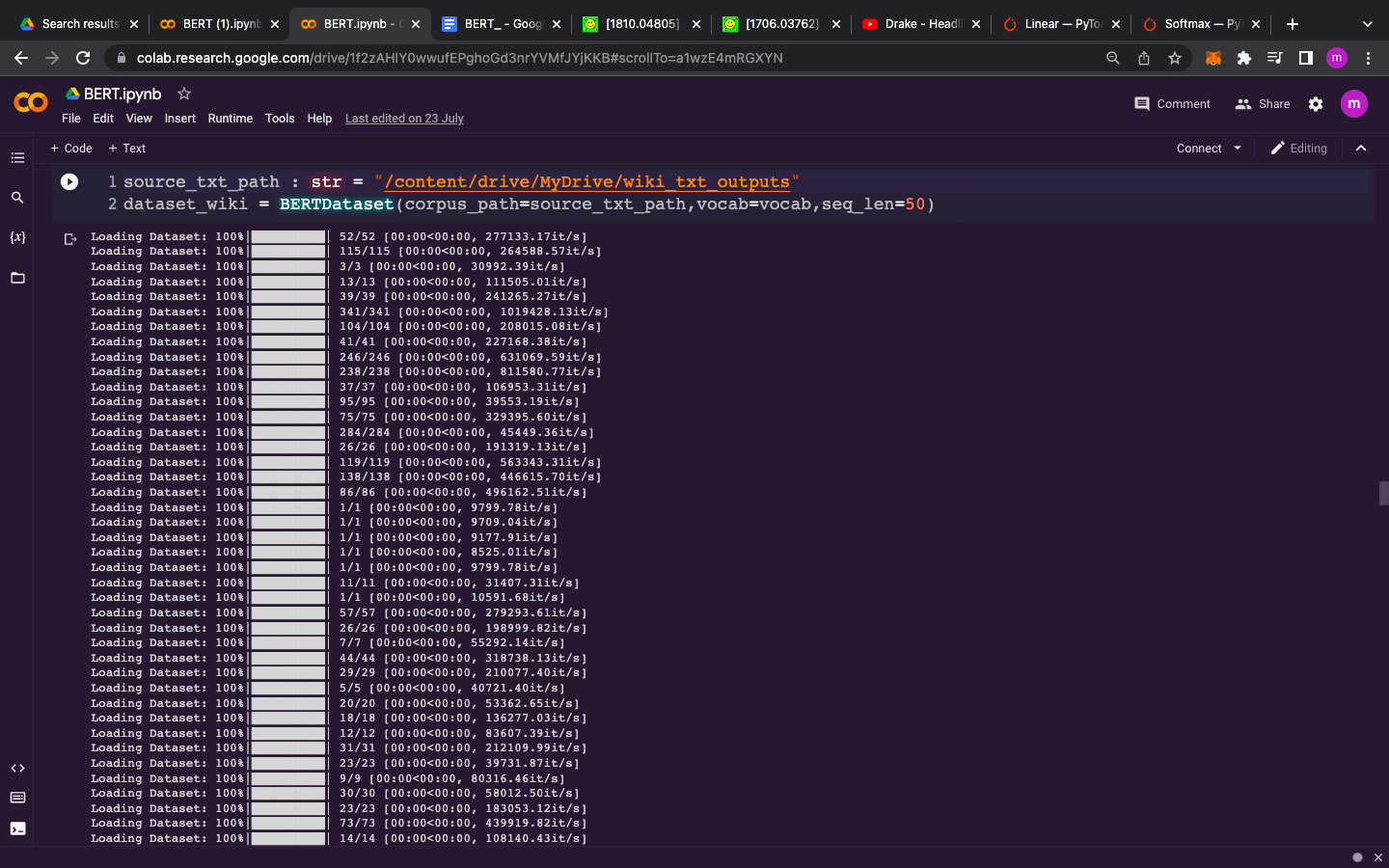


Figure 10 - Loading the Custom Data-set

To make BERT handle a variety of down-stream tasks, input representation proposed in the paper is able to unambiguously represent both a single sentence and a pair of sentences in one token sequence.

Original BERT uses WordPiece embedding with a 30K token vocabulary. Whereas due to technical limitations we used WordPiece embedding with 10K token vocabulary.

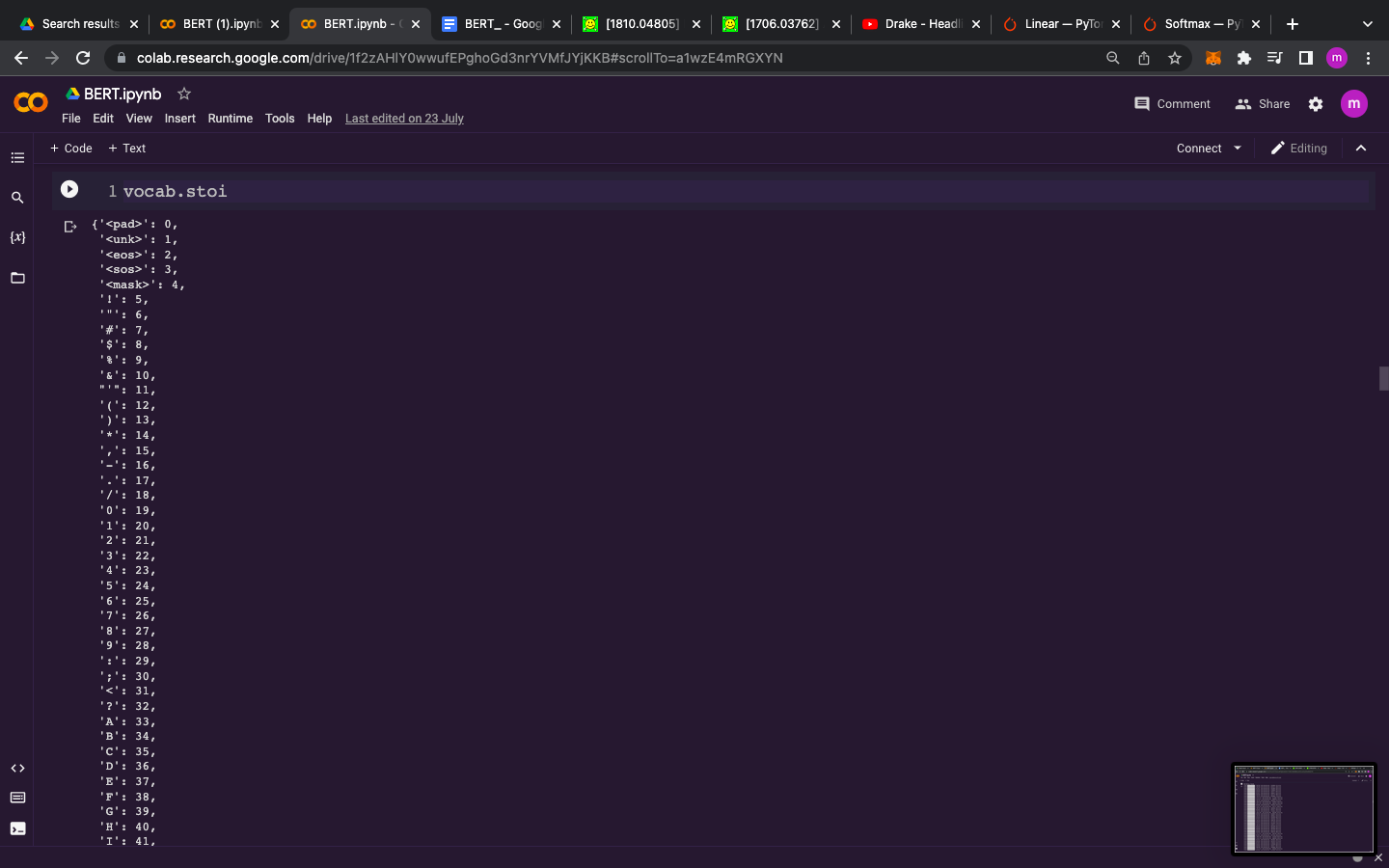


Figure 11 - 10,000 vocabulary built using WordPiece

The first token of every sequence is always a special classification token ([CLS]). The final hidden state corresponding to this token is used as the aggregate sequence representation for classification tasks.

The sentences are differentiated in two ways. First, we separate them with a special token ([SEP]). Second, we add a learned embed- ding to every token indicating whether it belongs to sentence A or sentence B.

BERT is trained unsupervised on the following two tasks :

1. Masked token prediction (MLM) : 15% of the input tokens are masked randomly and the model is trained to predict the original tokens. This helps the LM to reach deep bidirectional representation.
2. Next token prediction (NSP) : Many NLP tasks are based on understanding the relationship between two sentences. When choosing the sentences A and B for each pre-training example, 50% of the time B is the actual next sentence that follows A (labeled as IsNext), and 50% of the time it is a random sentence from the corpus (labeled as NotNext).

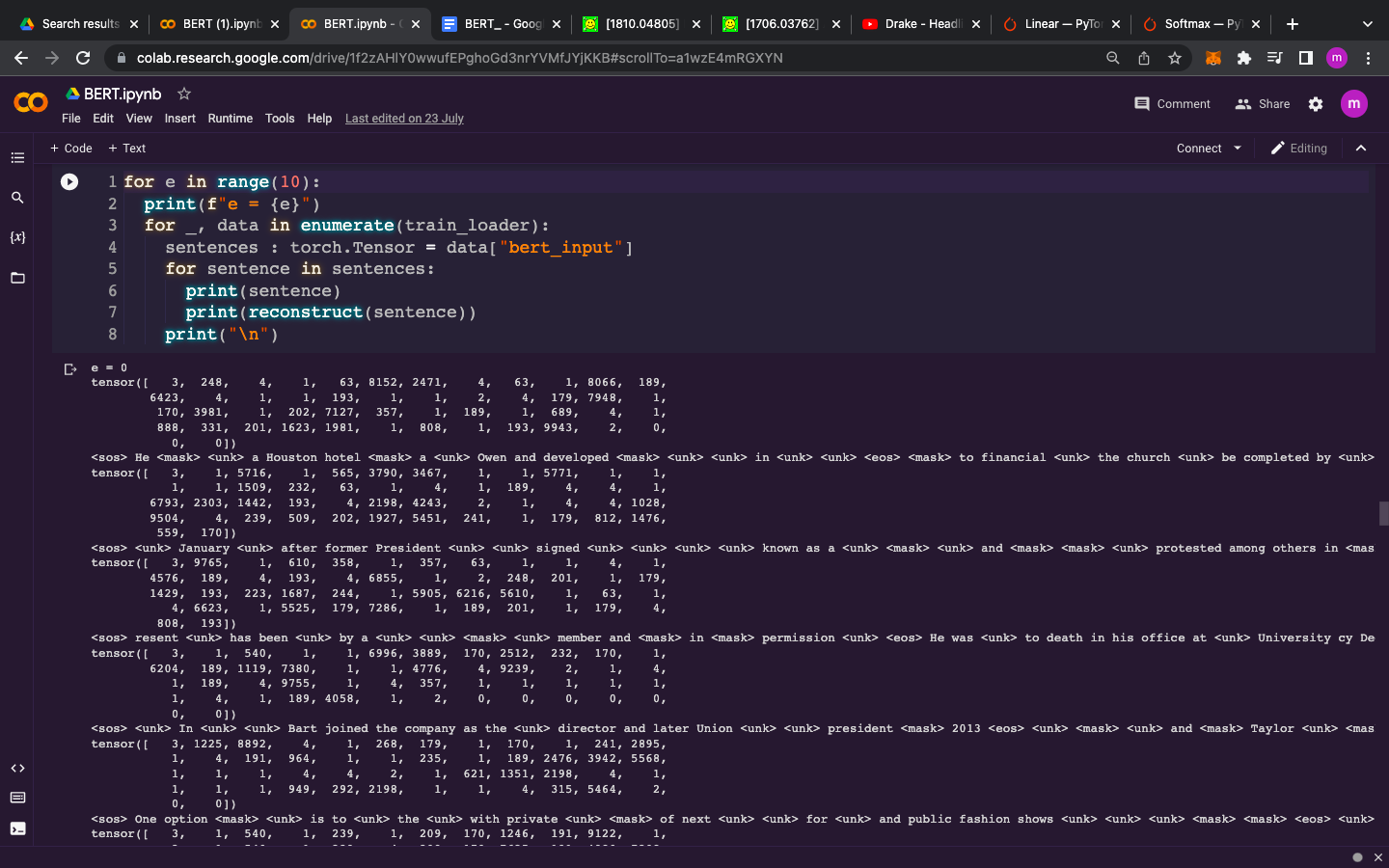


Figure 12 - Preprocessed Data

In order to feed the sentences into the LM, 3 types of embedding is performed on the sentences are the results are added before they are fed into the LM.

* Token embedding : WordPiece embedding with 10K token vocabulary was used. To perform WordPiece embedding “BertWordPieceTokenizer '' package from hugging face was used.
* Segment embedding : There are just two vector representations in the Segment Embeddings layer.All tokens belonging to sentence 1 are assigned to the first vector (index 0), whereas all tokens belonging to sentence 2 are assigned to the second vector (index 1).
* Positional embedding : The absolute position embedding is used to model how a token at one position attends to another token at a different position.

To learn the above mentioned embedding, we created a learnable embedding layer by inheriting nn.Embedding class from PyTorch.

**3.3.2 Implementation**

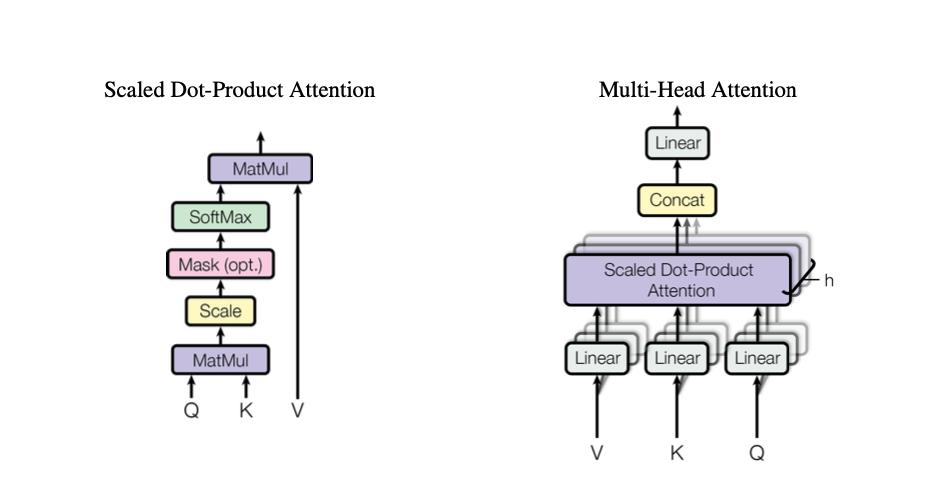


Figure 13 - Attention Architecture

BERT was built using a transformer's encoder. Rather than using existing packages, we implemented the transformer from scratch by implementing the paper “Attention Is All You Need”.

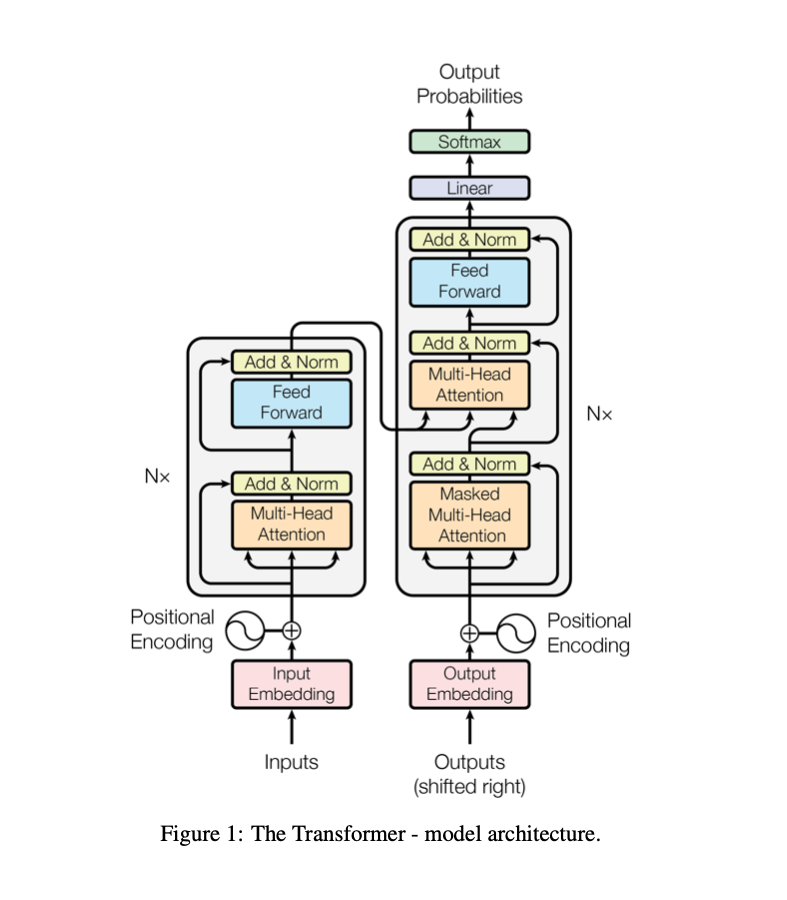
****

Figure 14 - Transformer Model

Below are the components of transformer explained :

* Attention : An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. [2]
* Multi-headed attention : Multi-head attention allows the model to jointly attend to information from different representation sub-spaces at different positions.

Each of the layers in our encoder and decoder of the transformer contains a fully connected feed-forward network, which is applied to each position separately and identically.

BERT’s model architecture is a multi-layer bidirectional Transformer encoder based on the original implementation in [2].

Transformer summary :

(encoder): Transformer(

(layer): ModuleList(

(0): BertLayer(

(attention): Attention(

(self): SelfAttention(

(query): Linear(in\_features=768, out\_features=768, bias=True)

(key): Linear(in\_features=768, out\_features=768, bias=True)

(value): Linear(in\_features=768, out\_features=768, bias=True)

(dropout): Dropout(p=0.1, inplace=False)

)

(output): SelfOutput(

(dense): Linear(in\_features=768, out\_features=768, bias=True)

(LayerNorm): LayerNorm((768,), eps=1e-12, elementwise\_affine=True)

(dropout): Dropout(p=0.1, inplace=False)

)

)

(intermediate): Intermediate(

(dense): Linear(in\_features=768, out\_features=3072, bias=True)

(intermediate\_act\_fn): GELUActivation()

)

(output): Output(

(dense): Linear(in\_features=3072, out\_features=768, bias=True)

(LayerNorm): LayerNorm((768,), eps=1e-12, elementwise\_affine=True)

(dropout): Dropout(p=0.1, inplace=False)

)

)

With our current limitation to GPU access, we implemented BERT small (L=12, H=768, A=12, Total parameters = 110M), where

L : Number of transformer blocks,

H : Hidden size,

A : Number of attention self-attention,

Our BERT was trained on a GPU provided by Google Colaboratory for 300 steps for an hour. It was able to achieve min(loss) of 3.05.

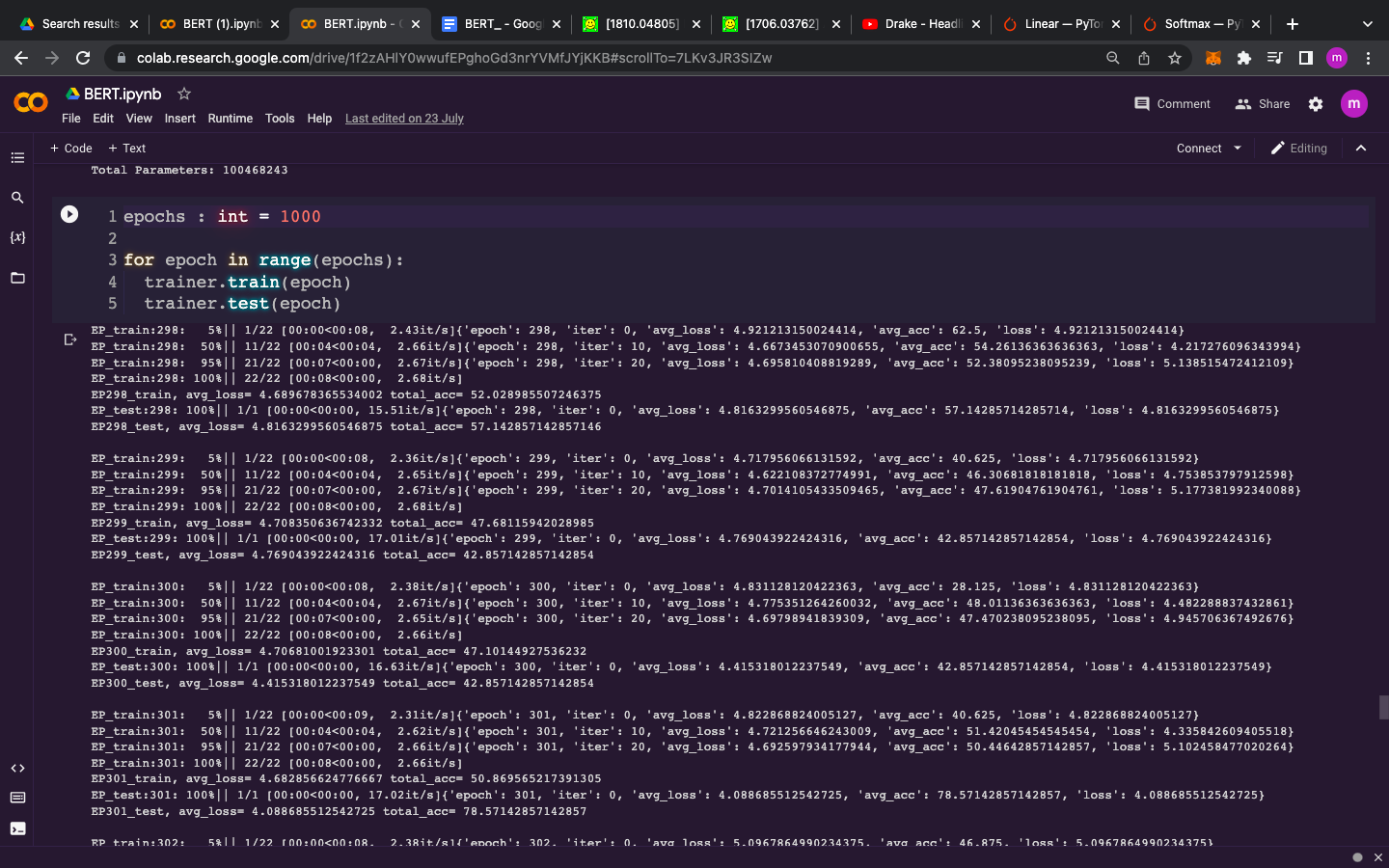


Figure 15 - BERT Training

**3.3.3 Classifier**

When we started to fine-tune the BERT from scratch, we ran into a problem. Tensor shape mismatch while feeding the output and label into the criterion. Due to time constraints, we left this problem unsolved. Therefore we used pre-trained BERT from hugging face and fine-tuned it to classify sentences into hate speech, offensive or normal.

We used a hate speech data-set from hugging face to train this classifier. Pre-processing mention in Data processing were used (WordPiece tokenization, position embedding and segment embedding)

The linear stack followed by BERT layer contains the following :

* Dropout layer : Linear layer with 768 input features and 3 output features to perform linear transformation.
* Softmax to rescale the tensor into n-dimensional output Tensor lies in the range [0,1] and sums to 1.

The BERT classifier was fine-tuning for 3 epochs on the same GPU described in the above section.

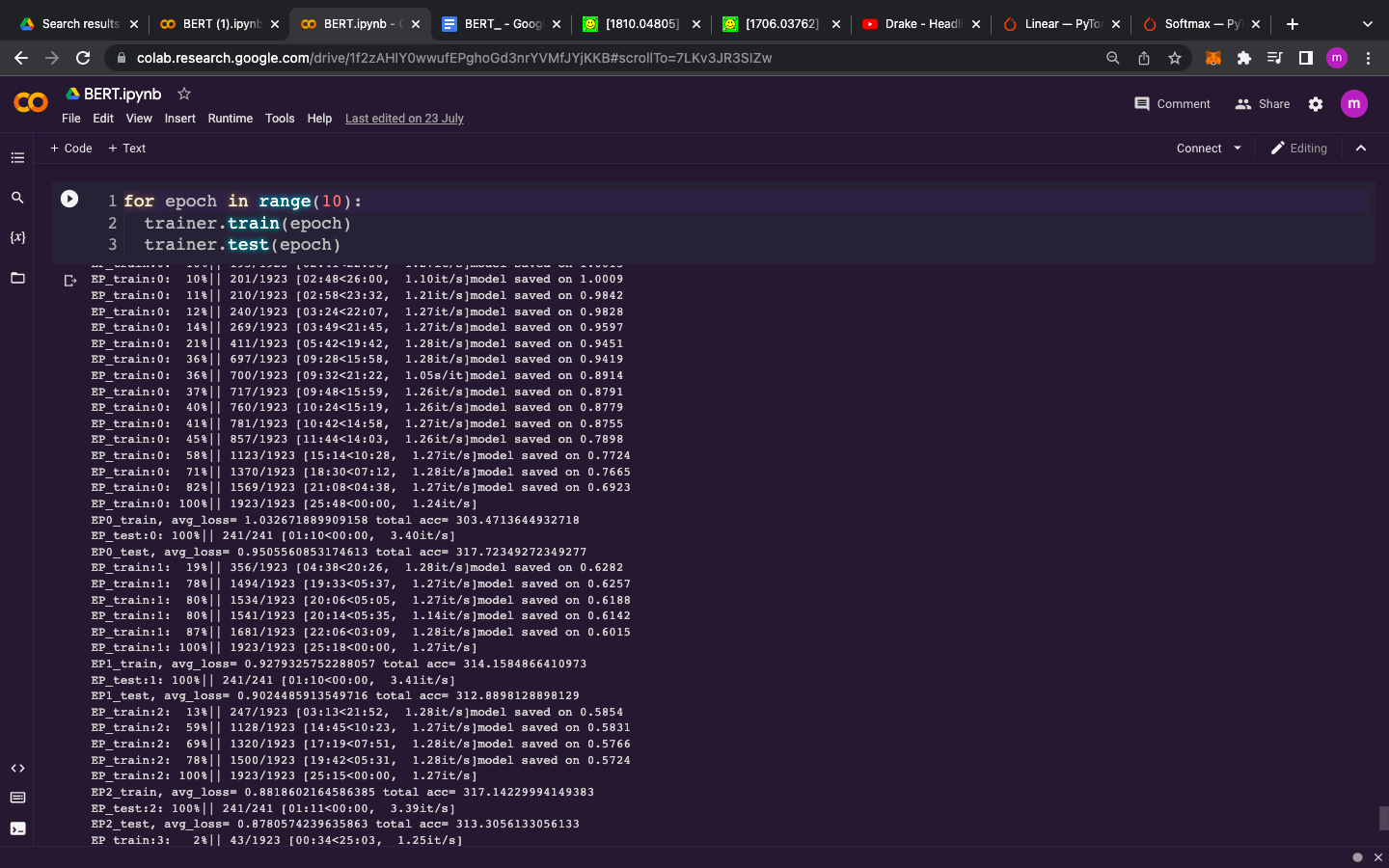
****

Figure 16 - BERT Classifier Training

**3.4 API Integration**

The model is deployed on an EC2 instance successfully and the API can be built in the instance itself using python flask library. Since the probabilities of hate speech and normal content is the only data returned by the model, a simple GET request from the created API with the content of a text message as input, will be sufficient.

Any application or platform can make use of this API to implement moderation. As an example, we’ve made use of the Discord.js library to create a bot that can perform auto moderation on messages being sent in a server. The Discord API sends info about event the bot can possibly can listen to : message creation, message edits, nickname changes and so on.

When the bot receives a message create event, it could send the content of the message to the API and receive the probabilities as JSON. The bot can make use of the response to either log the offensive content (so moderators can review) or delete the message by itself, if given the permissions to do. This way cyberbullying can be checked in a discord server through a discord bot.

**4. Technology used**

**4.1 Git and GitHub**

Git is a distributed version-control system. Version Control Systems are the software tools for tracking/managing all the changes made to the source code during the project development. It keeps a record of every single change made to the code. It also allows us to turn back to the previous version of the code if any mistake is made in the current version. Without a VCS in place, it would not be possible to monitor the development of the project.

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere. This tutorial teaches you GitHub essentials like repositories, branches, commits, and Pull Requests.

**4.2 PyTorch**

PyTorch is an open source machine learning library used for developing and training neural network based deep learning models. It is primarily developed by Facebook’s AI research group. PyTorch can be used with Python as well as a C++. Naturally, the Python interface is more polished. Pytorch (backed by biggies like Facebook, Microsoft, SalesForce, Uber) is immensely popular in research labs. Not yet on many production servers - that are ruled by frameworks like TensorFlow (Backed by Google) -Pytorch is picking up fast.

**4.3 Amazon EC2**

Amazon Elastic Compute Cloud (EC2) is a part of Amazon.com's cloud-computing platform, Amazon Web Services (AWS), that allows users to rent virtual computers on which to run their own computer applications. EC2 encourages scalable deployment of applications by providing a web service through which a user can boot an Amazon Machine Image (AMI) to configure a virtual machine, which Amazon calls an "instance", containing any software desired. A user can create, launch, and terminate server-instances as needed, paying by the second for active servers - hence the term "elastic". EC2 provides users with control over the geographical location of instances that allows for latency optimization and high levels of redundancy.

**4.4 AWS Lake Formation**

AWS Lake Formation is a service that makes it easy to set up a secure data lake in days. A data lake is a centralized, curated, and secured repository that stores all your data, both in its original form and prepared for analysis. A data lake lets you break down data silos and combine different types of analytics to gain insights and guide better business decisions.

**5. Results and Output**

**5.1 Data analysis on Cyberbullying**

We conducted our own analysis of cyberbullying using a data-set from kaggle. Gaussian Naive Bayes, Multinomial Naive Bayes and K neighbors classifier were training on the data-set. The following results shown in Fig. 17 and Fig. 18 were obtained :

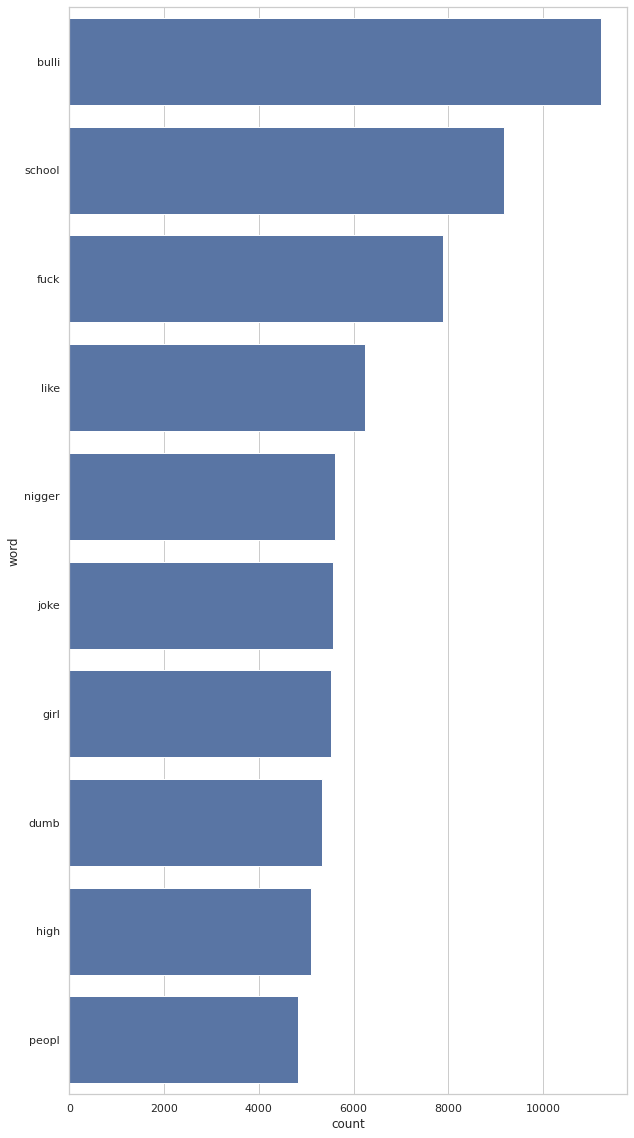


Figure 17 - Most Used Hate Words

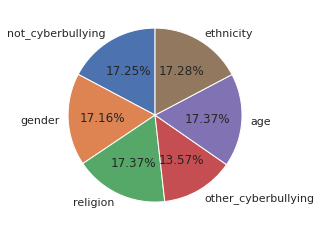


Figure 18 - Forms of Cyberbullying

**5.2 Model training results**

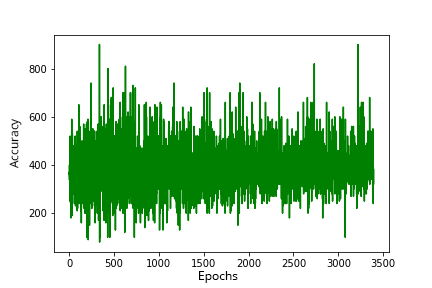


Figure 19 - Classifier Accuracy

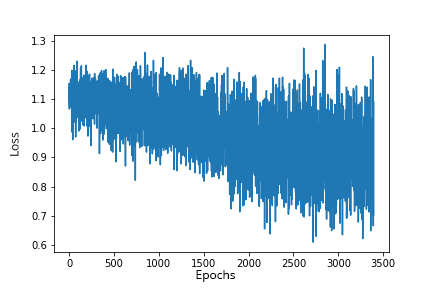


Figure 20 - Classifier Loss

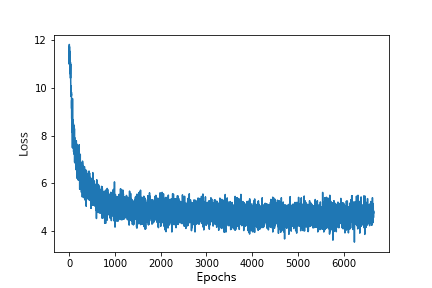


Figure 21 - Language Model Loss

**5.3 API Integration Example with Discord.js**

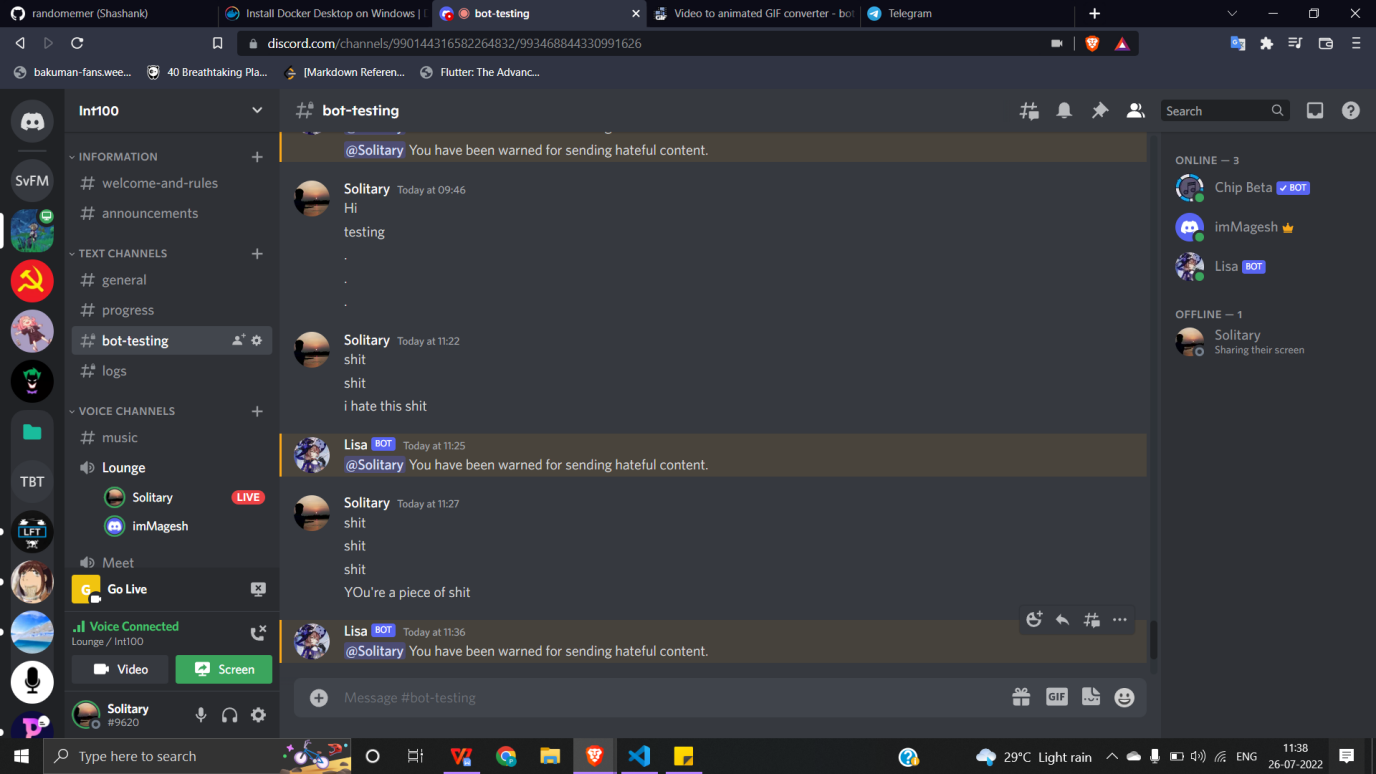


Figure 22 - Deleting messages and warning users

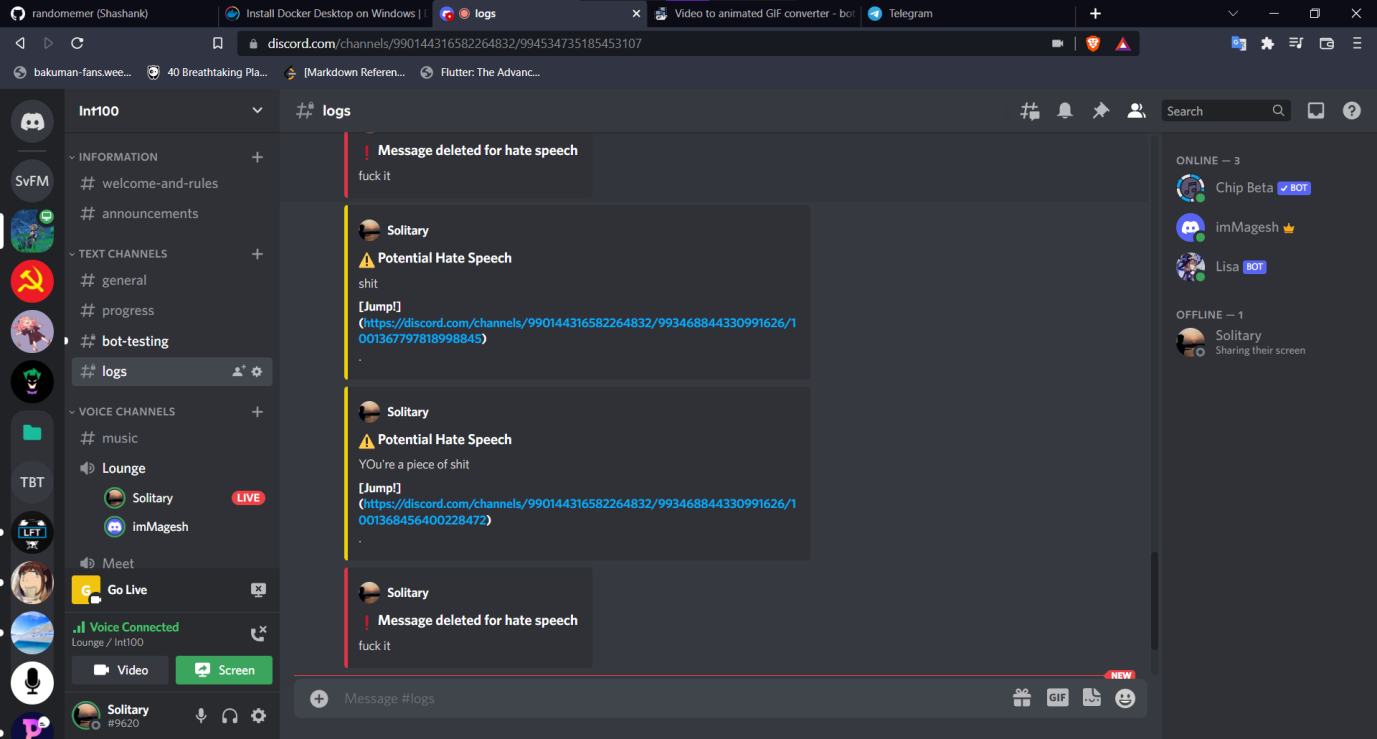


Figure 23 - Logging Cyberbullying related events

**6. Timeline**

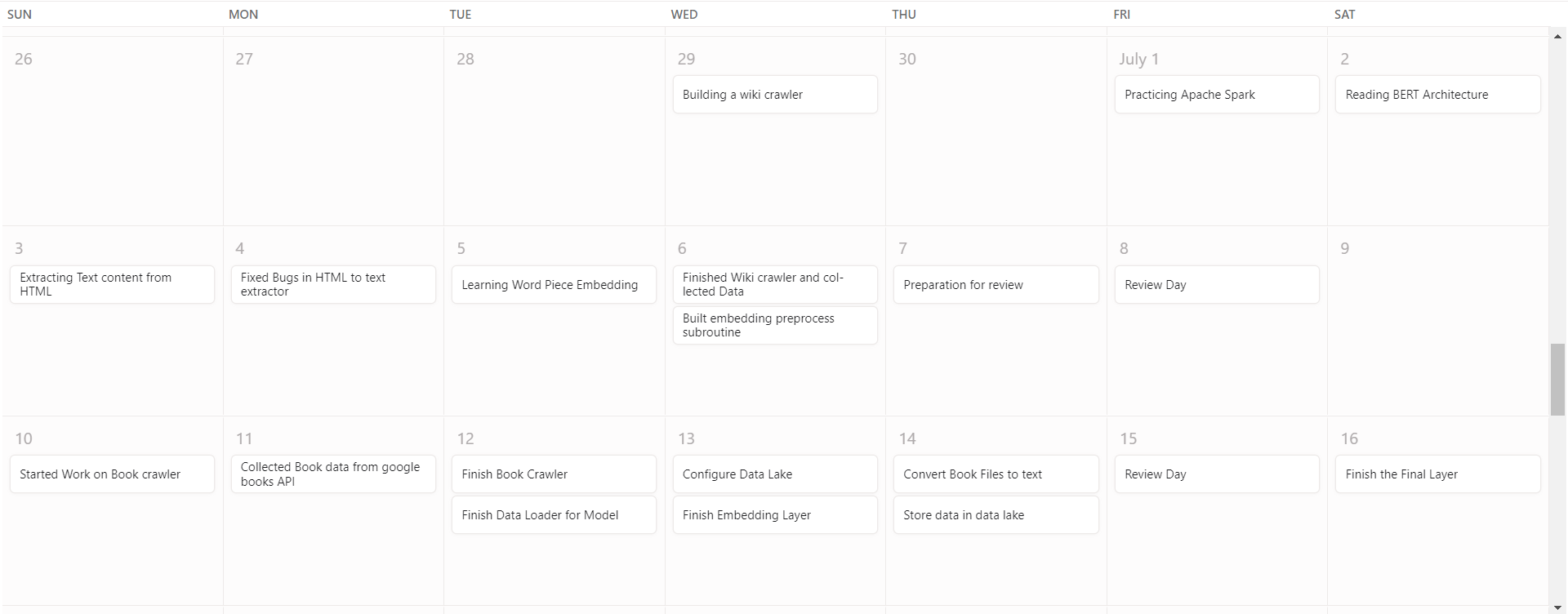
****

Figure 24 - Timeline June 29 - July 16

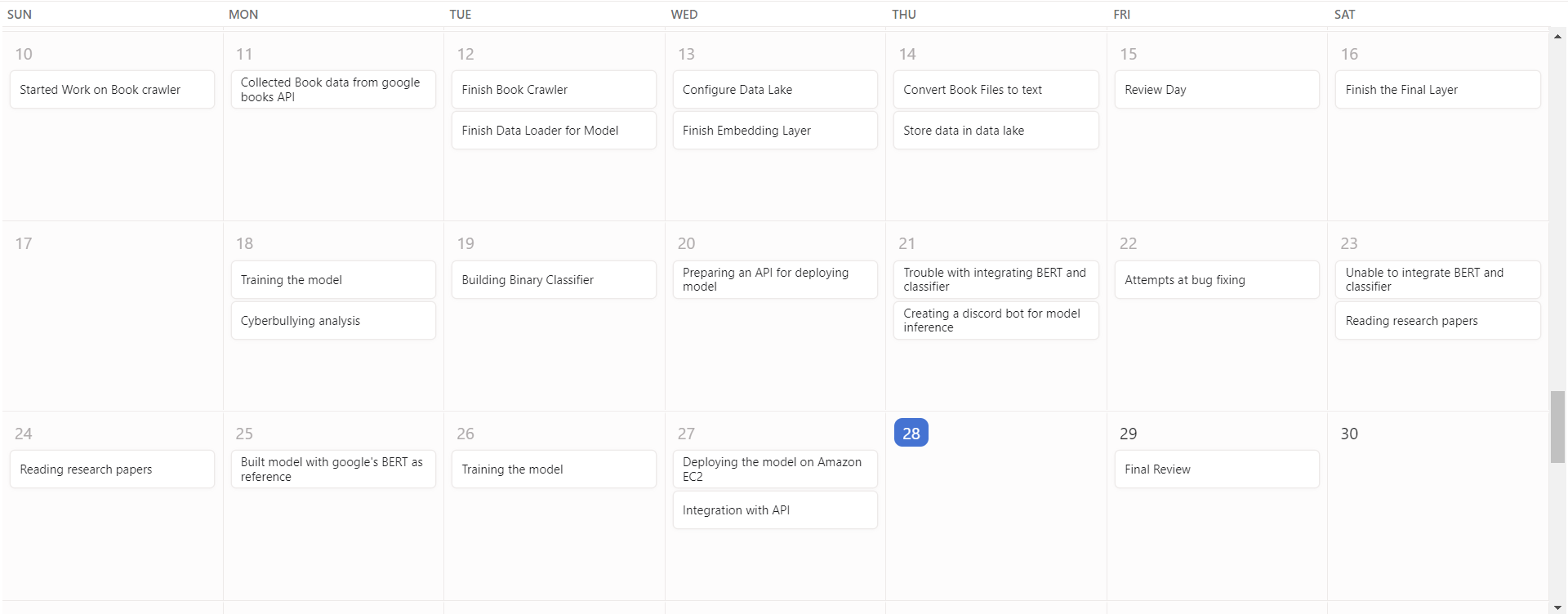
****

Figure 25 - Timeline July 10 - July 28

**7. Scope for further enhancement**

There’s always scope for improvement, thus there are ways to improve the performance and reliability of the model in real situations.

Firstly, training the model with more data will result in an accurate model when compared to those with weak training and fewer epochs. More data could be collected for this purpose. The crawlers could be allowed to crawl for longer periods of time, messages from platforms like discord and twitter could be collected and utilize data-sets which are available.

Secondly, the model integration with the classifier could be improved to obtain accurate and precise outputs. The model’s training could also be improved by increasing the number of parameters. As mentioned earlier, google’s BERT has a very large number of parameters. However, we should keep in mind that requires intense compute and GPU, hence they might have to procured by renting GPU online / buying physically.

Thirdly, the model could be also be extended to handle images and perhaps videos through computer vision for detection of cyberbullying. This will be truly resourceful for social media platforms as 3.2 billion images and 720,000 hours of video are shared online daily.

**Conclusion**

Cyberbullying is a prevalent threat on the internet and can be detrimental to the psychological health of those are victims of it. With the advent of technology and development, Cyberbullying can be combated through machine learning and artificial intelligence.

We built a model with BERT and classifier to detect hate speech in text content. This model was deployed to Amazon EC2 along with an API to access the model. This model can be integrated in application through the API to keep cyberbullying in check.

Through hate speech detection, it’s easier to tackle cyberbullying taking place in online social interactions. It helps keep discussions and conversations civil and prevents the harmful effects of online cyberbullying which can possibly cause damage to mental health. This will ultimately make online social platforms a little bit safer for everyone.

Providing a common way to make use of the machine learning model will make it easily accessible to anyone who wishes to integrate this in their projects. It’s taking a step towards a safer and better internet.

**References**

1. <https://www.unicef.org/end-violence/how-to-stop-cyberbullying>
2. <https://blog.ipleaders.in/cyber-bullying-complete-analysis/>
3. <https://www.kaggle.com/datasets/andrewmvd/cyberbullying-classification>
4. <https://www.wikipedia.org>
5. <https://discord.com/developers/docs/reference>
6. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding : <https://arxiv.org/abs/1810.04805>
7. Attention Is All You Need : [https://arxiv.org/abs/1706.03762](5.https:/arxiv.org/abs/1706.03762)
8. PyTorch : [https://pytorch.org/docs/stable/index.html](https://pytorch.org/docs/stable/index.html" \o "https://pytorch.org/docs/stable/index.html)