IJARIIT INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact Factor: 6.078

(Volume 7, Issue 3 - V7I3-2048)
Available online at: https://www.ijariit.com

Masking private user information using Natural Language Processing

Satwik Ram Kodandaram
satwikram29@gmail.com
Visvesvaraya Technological University, Bangalore, Karnataka

Kushal Honnappa
kushal.h1999@gmail.com
Presidency University, Bangalore, Karnataka

Kunal Soni
sonee069@gmail.com
Sanskar School, Jaipur, Rajasthan

ABSTRACT

The Internet brings a lot of efficiency to our lives but we must be aware that everyone exchanges a huge amount of data while interacting with the internet. One of the most important leisure activities one gets from the internet is to be able to socialize. For example, social media has become part and the core of our lives. With more than 2.3 billion active users, data privacy is an issue of concern. The world of the internet has become full of frauds hunting for personal information they leverage for their immoral activities. So, coming up with an algorithm that could secure data and process it such that no private data is involved, and machines continue to be trained with greater data. This could mean a dataset with data that is processed in a manner to make it anonymous. If there is any private information, we will mask that information with pseudo data. We use Named Entity Recognition using Deep Learning for identifying and masking personal information. In this paper, we will discuss how we mask private data. This model will be a successful technique to hide one's personal information to achieve complete data privacy.

Keywords: *Named Entity Recognition, Deep Learning, Social Media Platforms, Data privacy; Masking, Anonymity*

1. INTRODUCTION

Social media user's concerns about their data privacy have spiked in recent years. Incidents of data breaches have alarmed many users to rethink their relationships to social media and the security of their personal information. The dramatic story of consulting agency Cambridge Analytica is an example that exploited the private information of over 50 million Facebook users to influence the 2016 American presidential election. These issues are not acceptable with private user information. According to the study conducted by the Pew trust, 80 percent of social media user's information being concerned about businesses and advertisers accessing and using their social media posts. These privacy issues have prompted the advocacy of tighter regulations.

People will hesitate to share on social media as their data can be used or leaked. Given today's social media privacy issues and concern, skilled cybersecurity professionals will play a vital role in protecting the private user information and also the application needs some automation which can hide some private data.

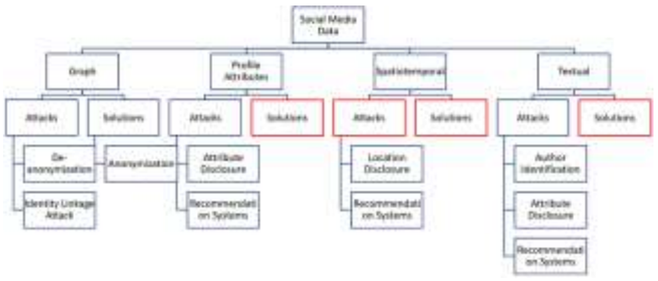


Fig. 1. An overview of privacy issues concerning the type of social media data Tasks

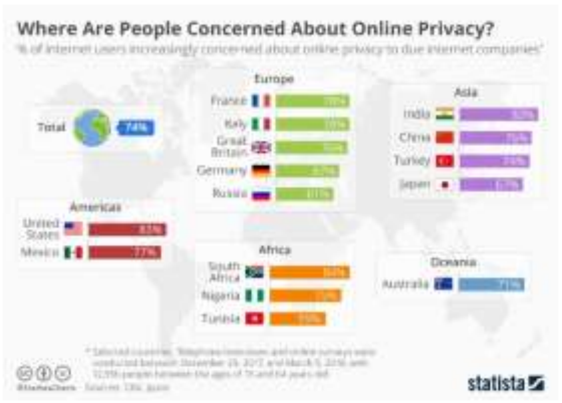


Fig. 2. Where are People concerned about Online Privacy

1.1 Motivation
Data privacy has become a very important topic in our lives as we are moving towards a more digitalized lifestyle. Crucial data is being collected of people when they are in different mental states through various digital devices installed in their homes or workplace. This data is later used to predict their behavior and maybe show advertisements accordingly. Even to train our machines for machine learning, we need to give them data, the data which is exposed to a larger group of people. But we are trying to anonymize this, anonymize the collected data, so even if it lands in the wrong hands, it holds no private information. This data can then be used for various other purposes as it still contains the gist of the whole idea.

1.2 Problem Statement
In the present scenario, we see a lot of people are facing mental health issues. This is mainly because of the stress which they face. This happens mainly due to the changed lifestyle of people. So mental health has become a need for all of us that needs to be addressed. Being physically fit is important, but at the same time, being mentally fit is also equally important for all of us. We have a lot of social media applications where users cannot share their thoughts/views.

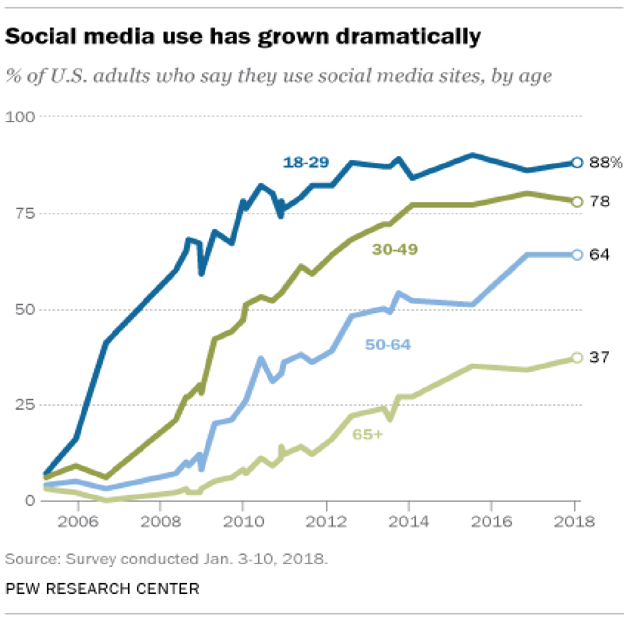


Fig. 3. Social Media usage growth

Below listed are some of the issues, why users cannot express their thoughts/views:
- ✓ Most of the social media application is collecting private information which can be hacked in the future and that information could be leaked to hackers who can misuse the data
- ✓ There is no complete user anonymity: Users will have to share their email with the website and then are allowed to post anonymously, which is not complete anonymity.
- ✓ Less accessibility: Most of these platforms are limited to the English language only. Moreover, the support for people who can't type is missing at most sites, which makes these platforms less accessible to the public.
- ✓ Targeted advertisements: Some platforms show targeted advertisements based on what users share. This could be a helpful feature, but this is something that is not beneficial when dealing with mental health. Users feel they are being tracked.
- ✓ Disclosure of private information on social media applications: Although users are allowed to post anonymously, sometimes, the data they share contains some private information about other people. This can lead to unexpected problems.

One of the solutions for the above-mentioned problems is an application where there is complete user anonymity. Complete user anonymity can be achieved by masking user's private information with pseudo data. We train Deep Learning algorithms to identify these private user's information and then mask this private information with pseudo data. We use Named Entity Recognition for identifying entities like Name, place, etc, and then we will mask it. By this, we can able to hide private user's information and achieve complete user anonymity.

1.3 Objectives
Our objective is to solve the problem of data privacy. We plan on removing the sensitive part from a given piece of data to ensure that no private information like names and addresses are present. All this data can be masked with pseudo data of the same category like mobile numbers, names, and addresses. This will keep the data legitimate as a whole and will also remove all the sensitive information it collects. This could be one of the methods to achieve complete data anonymity.

2. LITERATURE REVIEW
Currently, there are a lot of works that de-identify sensitive data and mask it with different approaches. This can be extended to different languages given a large dataset is available for the data:

2.1 Towards Personal Data Identification and Anonymization Using Machine Learning Techniques [1]
In this paper [1], they have implemented using Supervised Machine Learning Algorithms. If we use Deep Learning algorithms we can fine tune it and extend our model for another use cases using Transfer Learning concept but not possible using Machine Learning algorithms.

2.2 De-identification in Natural Language Processing [2]
This paper [2] focuses on the usage of NLP for de-identification and the importance it in different areas like medical, social media, and CVs and describes what data need to be preserved and removed.

2.3 An Introduction to NLP-based Textual Anonymisation [3]
This work by Ben Medlock [3] talks about building a corpus and the process of construction of the same. He critically evaluates the system and talks about the issues he faced working on the project. He also introduces the HMM-based tagger which could be used as a corpus for the anonymous data.

3. IMPLEMENTATION
3.1 Deep Learning
Deep Learning is a subfield of Artificial Intelligence where it replicates the human brain, human Neural System. Deep Learning can be supervised, semi-supervised or unsupervised. Deep Learning is capable of learning unsupervised from data that is unstructured or unlabelled. Deep Learning algorithms enable us to train machines and make machines to understand the data and make decisions based on the correlation that exists between the dataset, the same way how humans think to make decisions with billion neurons connections. Deep Learning models are very slow to train and require high computational power, nowadays GPU or TPU have become a requirement to execute the deep learning algorithms.

Although GPUs are very expensive yet without them training deep neural networks to high performance is practically not feasible. Today Deep Learning models are achieving the State of the Art (SOTA) on challenging machine learning problems like language translation from one language to another.

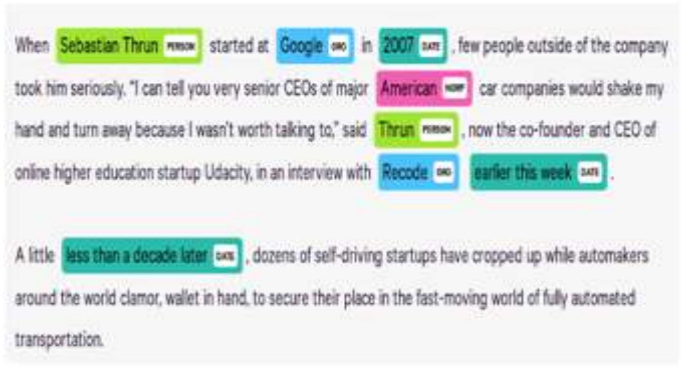
3.2 Natural Language Processing

A subfield of Artificial Intelligence is natural language processing. NLP's main goal is to understand and react to human languages. Like other Machine learning algorithms, NLP requires data to be trained. Working with text is very important and it is very hard as it requires knowledge from a diverse domain such as Linguistics, Statistics, Machine Learning, and these days Deep Learning. When data are trained with NLP models this algorithm tries to learn the language on its own with the help of available data its learning process is similar to humans learning natural languages. The advancement in NLP technology helps the world to grow better and faster. NLP helps humans in many aspects such as translation from one language to another this reduces the human's burden in knowing all the languages. NLP help's us to identify the tag of a given word, for example, "Bengaluru" with "geo-loc" as a tag.

The chatbot is an application of NLP that can be used in Customer support, Schedule a Meeting, Product Suggestions, Order Pizza, and so on. Usage of chatbots in these areas not only reduces workload but also saves customers waiting time. NLP has the skill that reads, writes, and also speaks the given languages the same as humans do. Combining all these techniques an application can be built that can behave, interact the same as humans without giving an impression of a machine.

3.3 Named Entity Recognition

Named Entity Recognition (NER) also known as Named Entity Identification, entity chunking, and entity extraction is a subtask of extraction of information from a corpus of sentences. It helps us to identify named entities like a person, location, event, organization, etc which are already pre-defined.

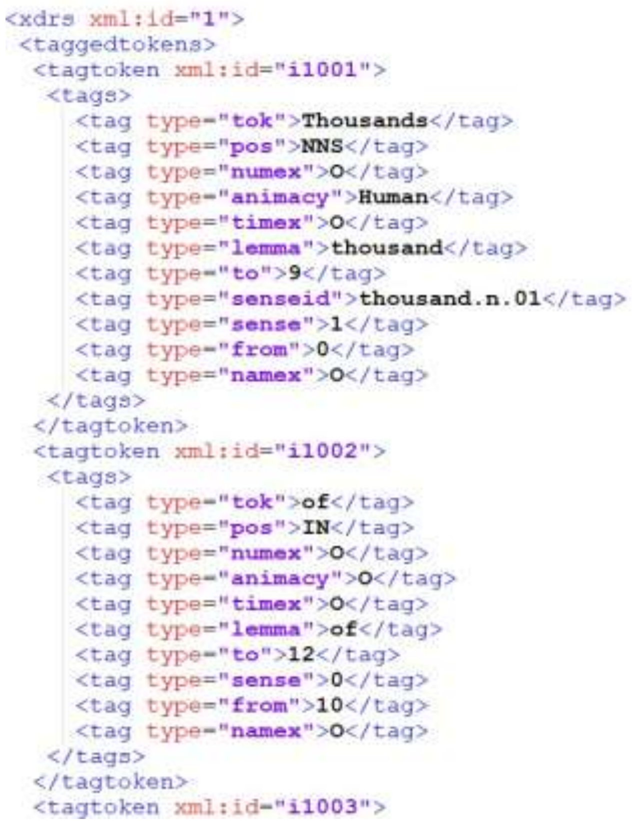


When Sebastian Thrun PERSON started at Google ORG in 2007 DATE, few people outside of the company took him seriously. "I can tell you very senior CEOs of major American NORP car companies would shake my hand and turn away because I wasn't worth talking to," said Thrun PERSON, now the co-founder and CEO of online higher education startup Udacity, in an interview with Recode ORG earlier this week DATE.

A little less than a decade later DATE, dozens of self-driving startups have cropped up while automakers around the world clamor, wallet in hand, to secure their place in the fast-moving world of fully automated transportation.

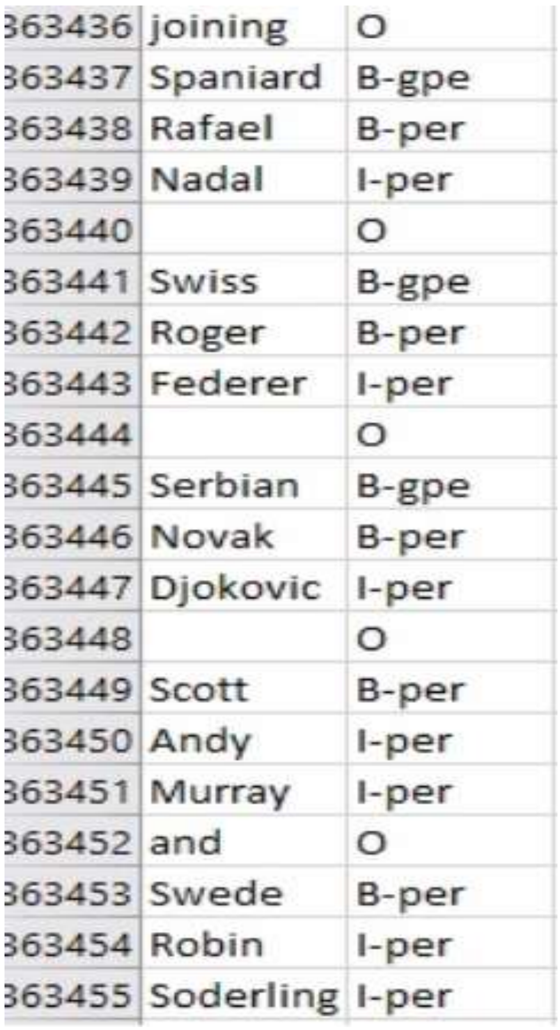
Fig. 4. Named Entity Recognition

Extracting main entities like a person, location, etc. helps us to sort unstructured data and detect important information, which is crucial if we have to deal with a large corpus of datasets. Named Entity Recognition can be achieved using Deep Learning classification where we give a set of training examples with labels stating words with the corresponding entity as labels/class to the model. On training Deep Neural Networks model with probability function can able to predict the class which word belongs to. There are a lot of applications on Named Entity Recognition, in our application, we use mainly named entity recognition to identify tags like a person, location, etc, and mask this information to hide the private information of the users and make our application complete user anonymous.



```
<xdrs xml:id="1">
 <taggedtokens>
  <tagtoken xml:id="i1001">
   <tags>
     <tag type="tok">Thousands</tag>
     <tag type="pos">NNS</tag>
     <tag type="numex">O</tag>
     <tag type="animacy">Human</tag>
     <tag type="timex">O</tag>
     <tag type="lemma">thousand</tag>
     <tag type="to">9</tag>
     <tag type="senseid">thousand.n.01</tag>
     <tag type="sense">1</tag>
     <tag type="from">0</tag>
     <tag type="namex">O</tag>
   </tags>
  </tagtoken>
  <tagtoken xml:id="i1002">
   <tags>
     <tag type="tok">of</tag>
     <tag type="pos">IN</tag>
     <tag type="numex">O</tag>
     <tag type="animacy">O</tag>
     <tag type="timex">O</tag>
     <tag type="lemma">of</tag>
     <tag type="to">12</tag>
     <tag type="sense">O</tag>
     <tag type="from">10</tag>
     <tag type="namex">O</tag>
   </tags>
  </tagtoken>
  <tagtoken xml:id="i1003">
```

Fig. 5. Data in XML



363436	joining	O
363437	Spaniard	B-gpe
363438	Rafael	B-per
363439	Nadal	I-per
363440		O
363441	Swiss	B-gpe
363442	Roger	B-per
363443	Federer	I-per
363444		O
363445	Serbian	B-gpe
363446	Novak	B-per
363447	Djokovic	I-per
363448		O
363449	Scott	B-per
363450	Andy	I-per
363451	Murray	I-per
363452	and	O
363453	Swede	B-per
363454	Robin	I-per
363455	Soderling	I-per

Fig. 6. XML to CSV

3.4 Data Collection and Data Pre-processing

3.4.1 Dataset: We have collected data from The Groningen Meaning Bank (GMB) which has a corpus of English texts with deep semantic annotations. The dataset has more than 1.4 million different tags, each tag representing different named entities with their corresponding words. GMB is an adequately large corpus with a lot of annotations. Unfortunately, GMB is not perfect. It is not a gold standard corpus, meaning that it's not completely human-annotated and it's not considered 100% correct. The corpus is created by using already existed annotators and then corrected by humans where needed. Here are the following classes in the dataset -

- ✓ geo = Geographical Entity
- ✓ org = Organization
- ✓ per = Person
- ✓ gpe = Geopolitical Entity
- ✓ tim = Time indicator
- ✓ art = Artifact
- ✓ eve = Event
- ✓ nat = Natural Phenomenon

The attached dataset is in tab-separated format; the goal is to create a good model to classify the Tag column. The dataset is labeled using the IOB tagging system.

The dataset is in XML form. The XML tree has words with all types mentioned.

We have converted XML form to a structured Data Frame, that is to CSV format. We have parsed over the XML tree, extracted the words and their corresponding tags, and converted them to CSV files.

3.4.2 XML to CSV Conversion: Let XML file be the Input that needs to be converted to CSV file. We will pass two empty lists that are words and tags list which need to be parsed and appended.

Let result be a list that consists of all the punctuations and other symbols which need to be filtered out from the XML file.

Algorithm 1: XML to CSV
Input: XML file, words list, tags list, and result
Output: Converted Xml to CSV

```
1.    Step 1: Parse the XLM tree
2.    tree ← parse(file)
3.    Step 2: Get root element from tree
4.    root ← tree.getroot()
5.    Step 3: Parsing throgh root and fetching words and tags
6.    for each elem in root, do
      for each subelem in elem.findall(tags), do
            if subelem attribute type == 'tok'
                  if subelem text in result
                        do nothing, pass or append(NaN)
                  else
                        words ← append(text)
            elif subelem attribute type == 'namex'
                  tags ← append(text)
                  end if
7.          end for
8.    end for
9.    Step 4: Create Data Frame from list
10.   df = DataFrame({"words": words, "tags":tags})
11.   Step 5: Convert Data Frame to CSV
12.   df.to_csv('ner.csv', index = 'false')
```

3.5 Tokenization

The process of splitting a sentence or a phrase into smaller units that will be individual words, this split piece of words is called as tokens. For example, let's take a sentence "Alex knows to program." on tokenizing we get ["Alex", "knows", "to", "program", "."] this tokenization [4] [5] is done by word boundaries. With the help of this tokenizer, we will be able to count the number of words in the text and count the frequently occurring words. After tokenization is done, we encode the tokens to numeric format.

3.6 Word Embeddings

Word Embeddings are created using neural networks with one input layer, one hidden layer and, one output layer. Neural Network doesn't understand the raw text given as input. We should encode it to the numbers using one-hot encoding or tokenization. Word Embeddings are like a numerical representation of a text. It is a type of word representation and learned representation that has words with similar meanings to have similar representation. Word Embeddings is a technique where individual words are represented as real-valued vectors which are predefined in the vector space of corpus. Each word is mapped to one vector and the vector values are learned in a way that resembles a neural network, which is more efficient while predicting.

Example of Predicting the next word:
Student Opened their _____
The corpus has houses, books, lamps, and stamps.

Here the prediction should be: Student Opened their books.

Numerical Representation of Words houses books lamps stamps
<0.6, 0. 2, 0.1, 0.1>

How it is represented in Word Embeddings Each row of W contains feature weights for the corresponding word in the vocabulary.

$$W = \begin{array}{c} \left(\begin{array}{ccc} 1.2 & -0.3 & 0.9 \\ 0.2 & 0.4 & -2.2 \\ 8.9 & -1.9 & 6.5 \\ 4.5 & 2.2 & -0.1 \end{array}\right) \end{array} \begin{array}{l} \text{books} \\ \text{houses} \\ \text{lamps} \\ \text{stamps} \end{array}$$

X will be the input given to the model to predict.

Each dimension of X corresponds to a feature to the prefix

$$X = < -2.3, 0.9, 5.4 >$$

We do Matrix Multiplication of W weights and input X to obtain W*X

$$W*X = < 1.8, -11.9, 12.9, -8.9 >$$

On top of W*X, a probability function is applied say softmax function for multi labels.
Softmax Function:

$$\text{Softmax}(X) = \frac{e^x}{\sum_i^n e^i} \qquad (1)$$

$$\text{Softmax}(W*X) = < 0.24, 0,73, 0.006, 0.002 >$$

The highest Probability is 0.73 that is for books so the prediction books.

Student Opened their books.

If the prediction is wrong, the Word Embeddings weights are adjusted to predict the right word.

3.7 Recurrent Neural Networks
We, humans, understand a sentence by understanding each word from it. A word is meaning changes concerning its past work. Similarly, neural network needs past event's information to understand now happening or future accruing events. This cannot be achieved by a conventional neural network.

This cannot be achieved by a conventional neural network. RNN [6] does this work, it does it by a network which is looped, this caries flow in information. An RNN is a duplicate of the same neural network where each one of it passes information to the other.

RNN works well when it needs information from a previous word from the same sentence. But it fails when it needs information from the previous sentence for example "Bangalore is capital of Karnataka."

In this sentence, Karnataka is predicted with the help of previous information. But in the sentence "Raju is a fisherman, so he catches fish daily" to predict the work fish there is a huge gap with the word fisherman. Here RNN fails to handle these "long-term dependencies". So, in these cases we use LSTM.

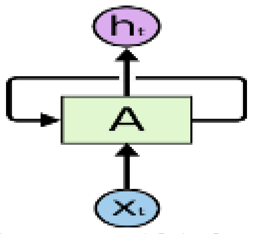


Fig. 7. RNN with its loops

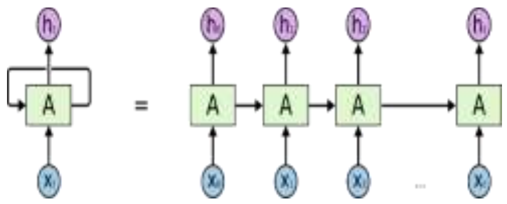


Fig. 8. RNN which shows that it is a combination of the same neural network

3.7.1 Long Short-term Memories (LSTM): LSTM [7] is a kind of RNN [6] that works well for long-term dependencies and also can remember the information for a longer period. It is capable of processing an entire sequence of data.

3.7.2 Difference between RNN and LSTM

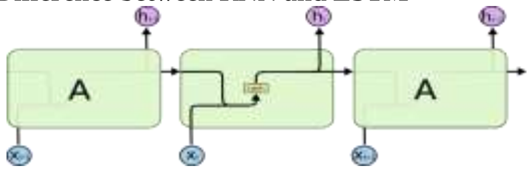


Fig. 9. RNN

In traditional RNN fig 9, we have only one tanh layer but in LSTM we have 4 interactions as shown in fig 10.

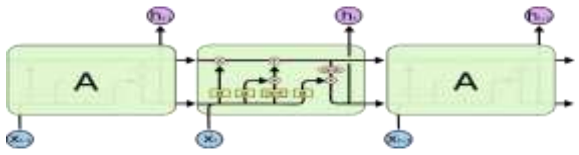


Fig. 10. LSTM

In fig 10 we can see there are many notations involved. Let's understand it by looking into fig 11.

Each line in fig 11 carries an entire vector from one node's output to the inputs of others. The learned neural network layers are represented by yellow boxes. The pink circle represents pointwise operations such as vector addition.

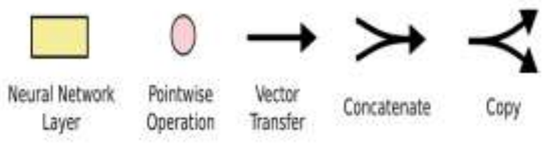


Fig. 11. Working LSTM

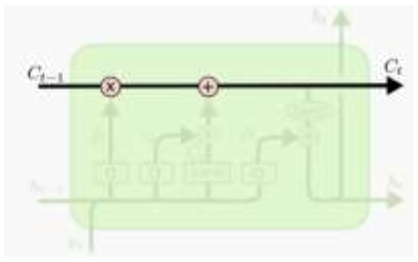


Fig. 12. Cell State

The alternation in the information like removing or adding the information is done using a structure called gates fig 13. These gates are made up of sigmoid neural network layers and pointwise multiplication operations.

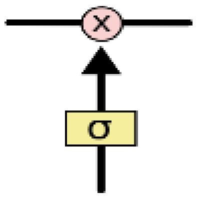


Fig. 13. Gates

This sigmoid layer gives output as 0 or 1. When it's 0 none of the information is left passed. If it's 1 then all the information is passed through it.

This works fine when we want to predict the future word or future event. But how can we predict the middle word for example fill in the blanks question? In this case, we need knowledge of the previous word and also the next word of the blank space. This cannot be solved using LSTM. Here comes the concept of Bidirectional LSTM. Let's see that is a Bidirectional LSTM (Bi-LSTM).

3.7.3 BI-LSTM (Bi-Directional Long Short-Term Memory)
Bidirectional LSTM [8] is a sequential model which consists of two LSTM [7]. Where one process in the forward direction and another process in the backward direction. This model not only helps to predict immediately following words but also precede word.

This model works well in our implementation as in our project we need to recognize the tags such 'O' tags this includes as shown in fig 2, person name, location name. To detect these tags the model needs information on both adjacent words for example in the sentence “Ram went to Mumbai” here to predict Ram is a name of a person it needs knowledge of the future word. Similarly, to predict Mumbai as a location name it needs knowledge of precede word.

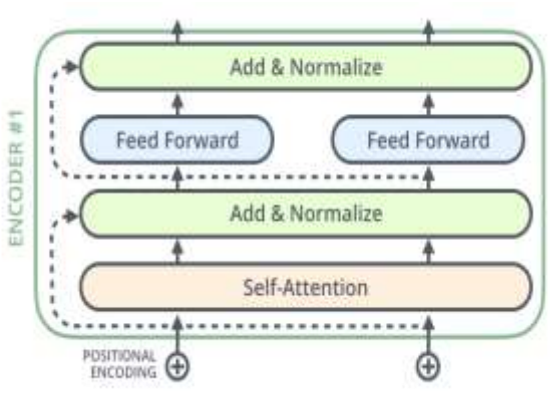


Fig. 14 Transformer

3.7.4 Transformers and Multi-head Self-Attention Mechanism

The Transformer was proposed by Google AI Team in the research paper [9]. The Transformer outperforms Google’s Machine Translation model in specific tasks. The biggest benefit of using a Transformer is how Transformer lends itself to parallelization. The Transformer a model that uses a self-attention mechanism to boost the speed at which neural network models can be trained. Self-attention can completely replace recurrence and helps to focus on particular words and their relationship with other words.

A transformer has several Encoders in it. The number of Encoders is a Hyper-Parameter. In the official Paper [9] Transformer has 6 Encoders. In Encoder we have, Positional Encoding, Self-Attention Layer, and Feed Forward Network with Residual Connections as shown in fig 10. On top of the Word Embeddings, we apply 3 different projections of linear layers vector space and obtain query, keys, and values. For every input word, we will apply linear layers and get Query, Keys, and Values. These Query, Keys, and Values come from the same text.

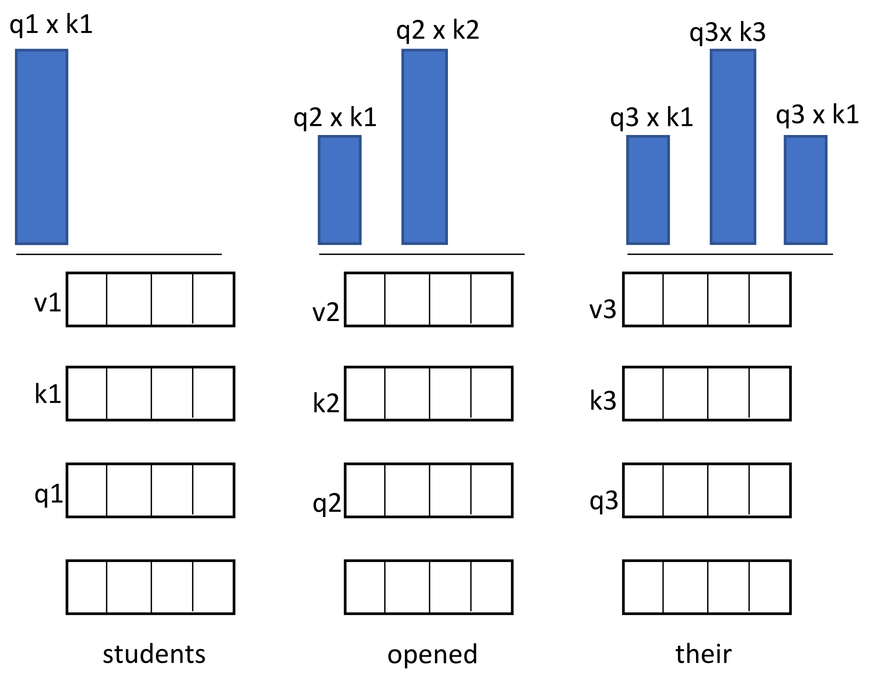


Fig. 15. Self Attention

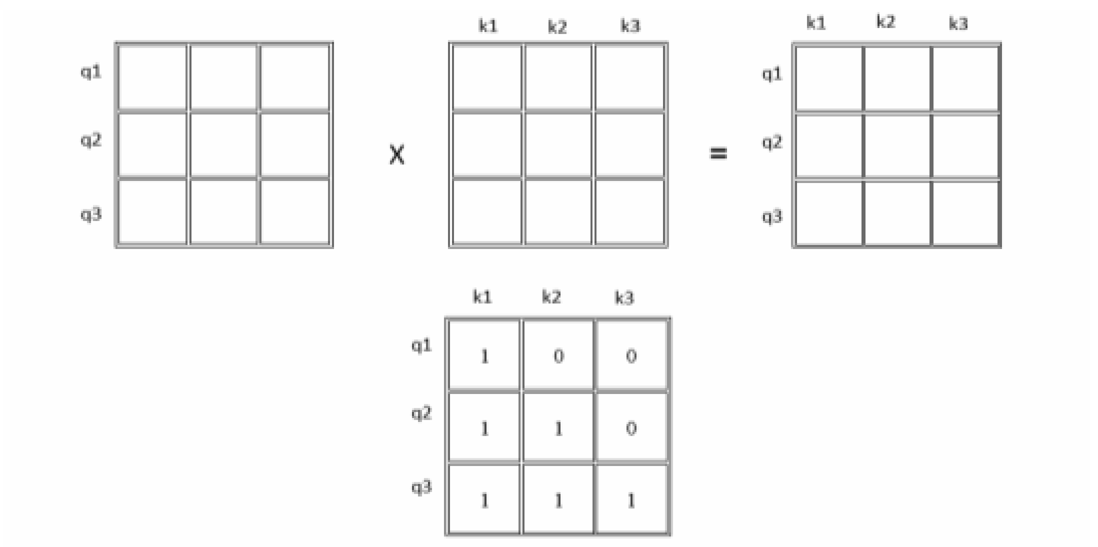


Fig. 16. Self Attention, Matrix Multiplication, and Masking

In the next step, we take the dot product of query and keys and obtain attention scores. The query and keys are used to compute the attention and values are used to compute attention-weighted representation.

For example, in predicting the next words, we obtain query, keys, and values for each word. On top of that, we obtain the probability. For the third word, we have distributed over all 3 keys. For the second word two distribution because here we don't include the third key in our attention weighted average as it is completely independent of the third time stamp and for the first word only one as mentioned in fig 16. Here in the matrix multiplication, we have masked with 0 but in reality, we will mask with some negative numbers.

We obtain z1, z2 and z3 as follows:

$$z1 = q1*k1 + v1 \quad (2)$$

$$z2 = q2*k1 + v1 + q2*k2 + v2 \quad (3)$$

$$z3 = q3*k1+v1+q3*k2+v2+q3*k3+v3 \quad (4)$$

This is how we get token-level representation. On top of these z1, z2 and z3 we apply SoftMax layer.

Here there is no dependency between zn and zn-1

Doing Parallel all attention computation by just matrix multiplication.

We will mask, after masking we apply SoftMax and we get valid attention distribution.

The same in the Transformer model we extend this to multiple heads. That means, we have many different projection matrixes and compute the attention. This operation is called a multi-head self-attention mechanism.

Many variants of attention are:

$$\text{Original } a(q, k) = w2T * \tanh(w1[q;k]) \quad (5)$$

$$\text{Bilinear Product: } a(q, k) = qTWk \quad (6)$$

$$\text{Dot Product: } a(q, k) = qTk \quad (7)$$

$$\text{Scaled dot Product: } a(q, k) = \frac{q^T k}{\sqrt{|k|}} \quad (8)$$

3.7.5 Positional Encoding: Attention models don't contain any recurrence or convolution, positional encoding is added to the model to give some information about the relative position of the words in a sentence.

This positional encoding is added to the embedding layer. Embedding represents a token in d-dimension space where tokens with similar meanings will be closer to each other in space.

But the Word embeddings don't give any relative positions of the words in a sentence. So, after adding positional encoding to the word embeddings, similar words will be closer to each other in the d-dimension space.

The formula to calculate the Positional Encoding is:

$$\mathbf{P_{i,j}} = \sin\left(\frac{\mathbf{i}}{\mathbf{10000^{\frac{j}{d_{emb-dim}}}}}\right) \text{ if j is even} \quad (9)$$

$$\mathbf{P_{i,j}} = \cos\left(\frac{\mathbf{i}}{\mathbf{10000^{\frac{j-1}{d_{emb-dim}}}}}\right) \text{ if j is odd} \quad (10)$$

3.7.6 Bert: BERT [10] is a trained Transformer Encoder stack. BERT has two variants, BERT Base with 12 Encoders and BERT Large with 16 Encoders. Transformer Encoders are the basic building blocks for BERT. The base for BERT is Transformer. Now we have open-source Pre-trained BERT available online, where we can change the last layer of the BERT and our custom function to perform our tasks.

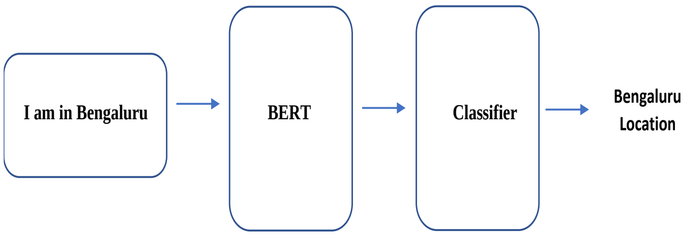


Fig. 17. BERT for Named Entity Recognition

Here we add our custom named entity recognition classifier on top of the BERT with SoftMax Layer using the transfer learning concept. We will include the top layers of the pre-trained BERT and we will train our model which will identify the named entities.

3.8 Proposed Model

We have collected the data from The Groningen Meaning Bank (GMB) which has a corpus of English texts with deep semantic annotations. It has around 1.4 million named entities. The data was in XML form, which we have converted from XML to CSV. The data has around 12 tags but we have taken only the essential tags that are required for our application.

After Data Pre-processing, we have tokenized the text and splitted the data into train and test. We are building different models using RNN, LSTM, Attention model, and BERT architecture and we will ensemble the predictions using Horizontal Voting to get a more generalized model.

This final model is used to predict the named entity like Name, location, etc. Once we identify these entities, we will mask this to some pseudo data.

Working Example:
Raw Text Input: "I am Satwik, I am in Bengaluru."
Output from the model: "I am XXX, I am in YYY."
This model is being deployed in Django Rest Frameworks with Flutter as frontend. By this user can post any data as data will be completely masked.

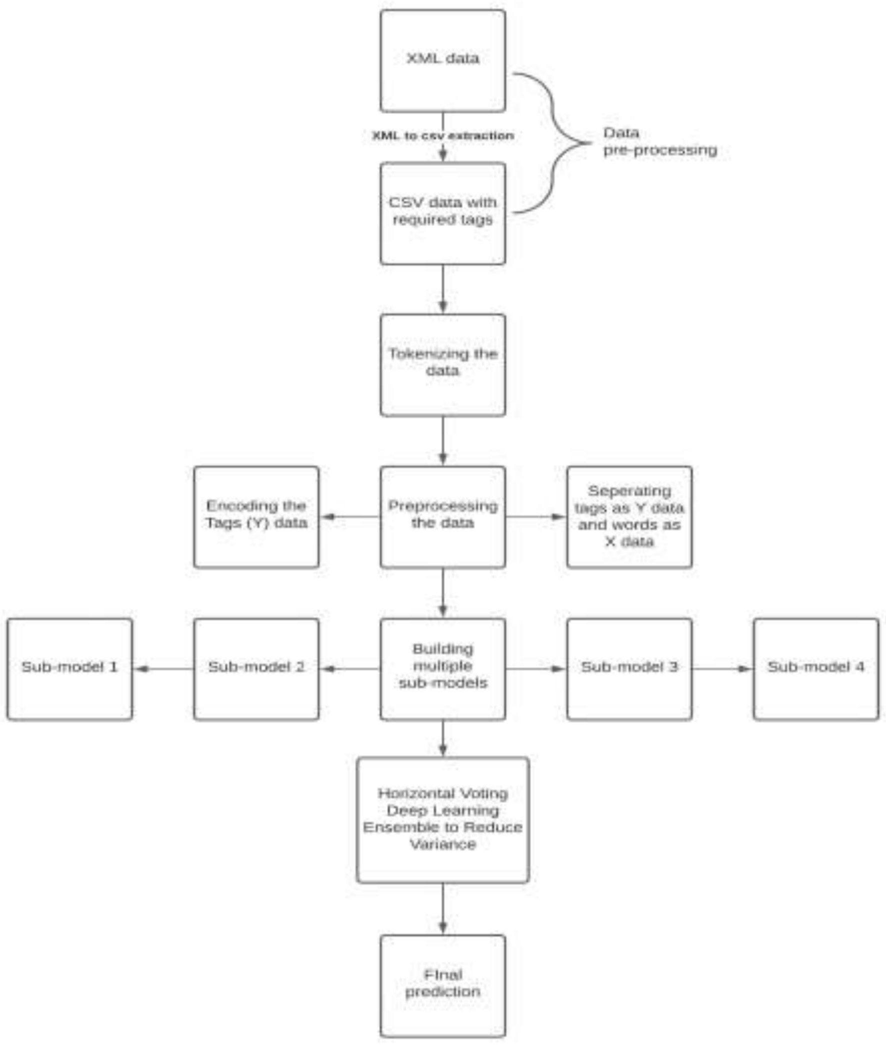

XML data
XML to csv extraction
CSV data with required tags
Data pre-processing
Tokenizing the data
Encoding the Tags (Y) data
Preprocessing the data
Seperating tags as Y data and words as X data
Sub-model 1
Sub-model 2
Building multiple sub-models
Sub-model 3
Sub-model 4
Horizontal Voting Deep Learning Ensemble to Reduce Variance
Final prediction

Fig. 18. Proposed Model

3.8.1 Horizontal Voting Deep Learning Ensemble to Reduce Variance with different models

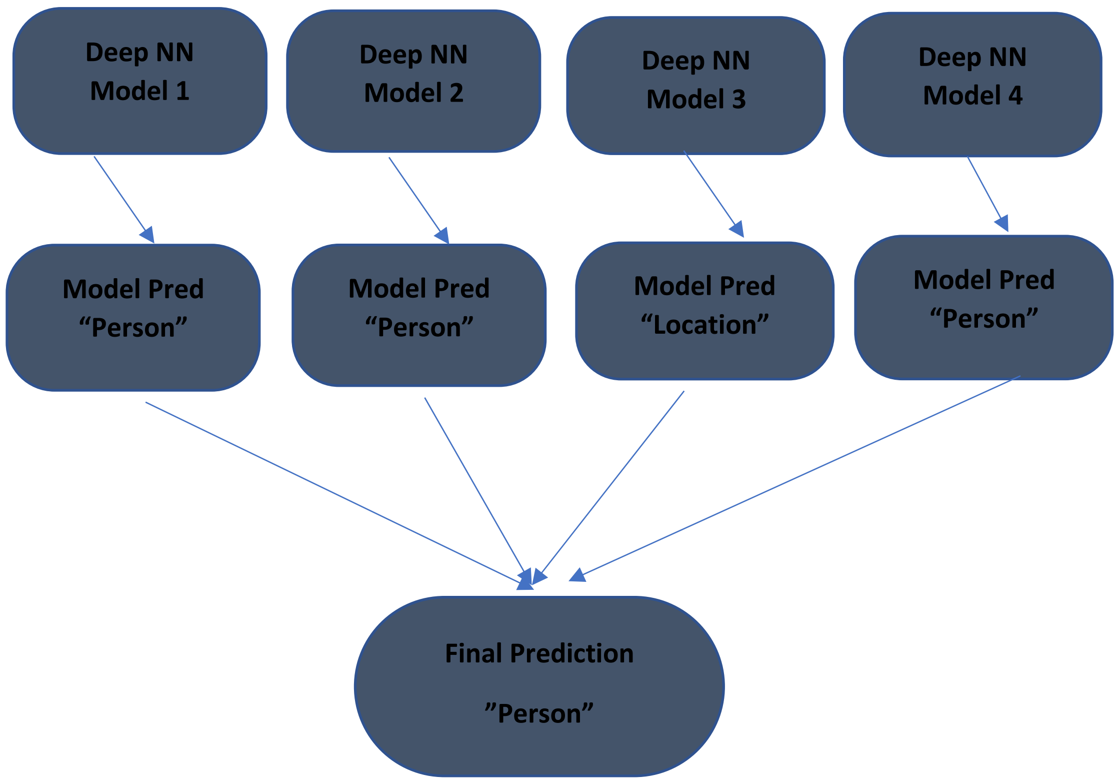


Fig. 19. Horizontal Voting Deep Learning Ensemble

Predictive modeling problems where we have a smaller number of training data relative to the number of unlabelled examples are challenging. Neural networks are used to train these kinds of problems and neural networks perform well on these types of the problem although they can suffer from high variance in model performance as measured on validation set or hold-out set.

This makes choosing the final model at end of the epoch is risky as there is no clear signal of which model is performing well compared to others towards the end of the training run.

As we have ensemble technique in Machine Learning [11] similarly we have Neural Networks Ensemble [12] [13] [14]. We have many ensemble techniques like Staking Generalization [15].

One of method in ensemble technique is the horizontal voting ensemble which is a simple approach to address this issue, where we have a collection of models saved, and these saved models are used as an ensemble that results in more stable and better performance on average compared to randomly choosing a single final model. This approach was developed specifically for those predictive modeling problems where the training dataset is small compared to the number of predictions to predict by the model.

In the above figure, we can see we have 4 Deep Neural Networks sub models each can have the same architecture or different architecture. Each sub-model predicts the probability of the class, here we can see prediction as "Person".

Here 3 sub-models predict as a person and 1 sub-model predicts as a location. So, the final model prediction is the "Person" class as it has 3 votes. Here, for example, we have taken 4 sub-models but in reality, we go with odd numbers of sub-models to avoid equal class prediction by sub-models.

Let model be the set of neural network models being trained on the training set T(xi,yi), such that m ∈ model. Let yhat be the predictions obtained by all the models on the test set T'(xi',yi'). Let 'array' be the function for converting lists to an array

Algorithm 2: Ensemble Prediction

Input: models, test set T'(x_i',y_i'), and empty yhat list

Output: predictions – final prediction obtained

```
1.      Step 1: Obtain the predictions of each model
2.      for each i in range($x_i$)
3.          for each m in model, do
        yhat[i] ← predict(x[i])
        Calculate highest number of votes for ith test data and append
        yhat[i] ← highest voted class
    end for
4.      end for
5.      Step 2 : Convert list into an array
6.      yhat ← array(yhat)
7.      Step 3: return yhat
```

4. EXPIREMENTAL RESULTS

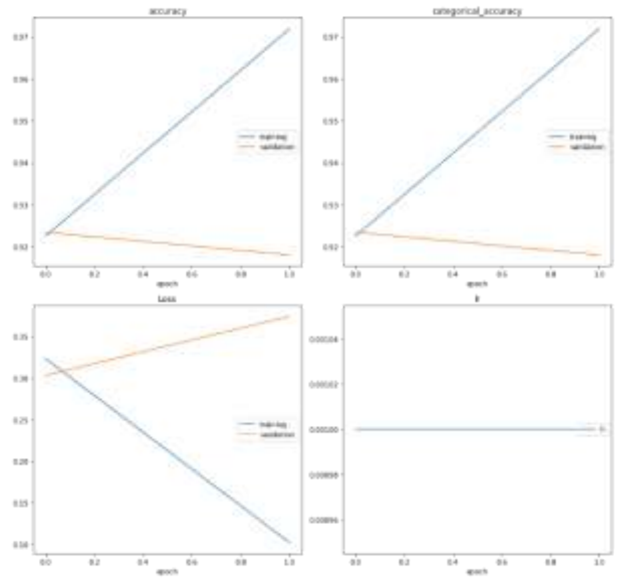


Fig. 20. Model Training plots

We have trained all the models for around 20-50 epochs with a batch size of 32.

We have done several experimented with our project by training our models with different algorithms, and with different approaches, we used BI-LSTM, Transformers, Bert. Here it was found that each model achieved good results.

For all the models while training we have used ReduceLROnPlateau callback with factor = 0.2, patience = 5, min_lr = 0.001 and Learning rate scheduler.

When a user enters a sentence like, "Ramesh lives in Bangalore." then the following result should be obtained "Ramesh" as name-tag, "lives" and "in" as o-tag, and "Bangalore" should be identified as loc-tag.

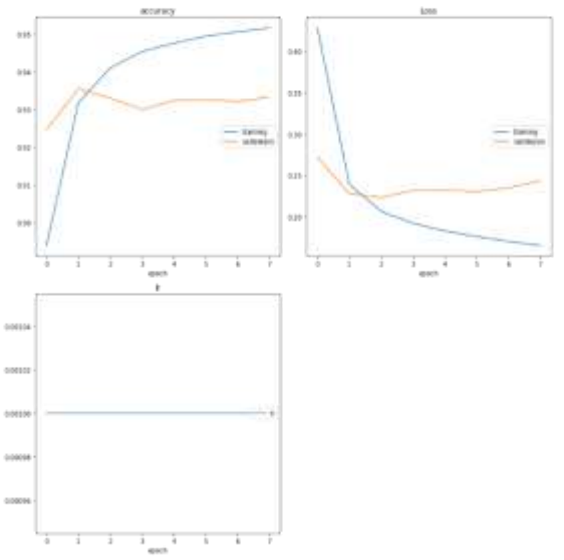


Fig. 21. Model Training plots

After achieving this we will be masking the information into the generalized form to provide anonymity for the users. For this example, we will be replacing name-tag and loc-tag with generalized form and only o-tags values will remain the same. The overall result will be formed as <person name> lives in <location name>. This can also be formed by creating a random character and replacing it instead of private data like "gtans lives in ytend" and explained as "gtans" is a person name and "yten" a location name. Here gtans and yten are the pseudo data.

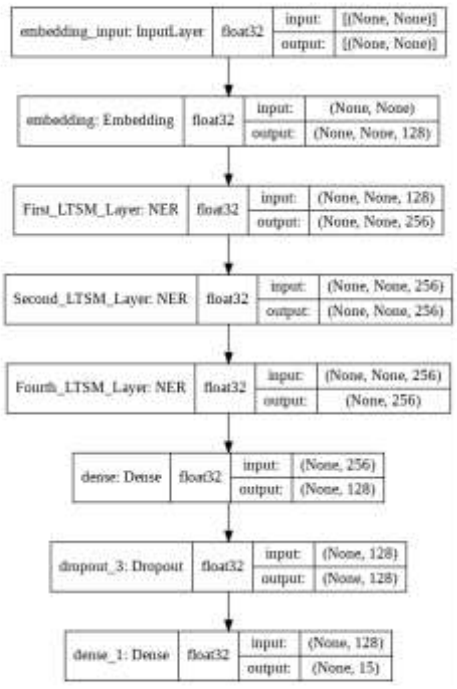

embedding_input: InputLayer | float32 | input: [(None, None)] | output: [(None, None)]
embedding: Embedding | float32 | input: (None, None) | output: (None, None, 128)
First_LTSM_Layer: NER | float32 | input: (None, None, 128) | output: (None, None, 256)
Second_LTSM_Layer: NER | float32 | input: (None, None, 256) | output: (None, None, 256)
Fourth_LTSM_Layer: NER | float32 | input: (None, None, 256) | output: (None, 256)
dense: Dense | float32 | input: (None, 256) | output: (None, 128)
dropout_3: Dropout | float32 | input: (None, 128) | output: (None, 128)
dense_1: Dense | float32 | input: (None, 128) | output: (None, 15)

Fig. 22. LSTM model

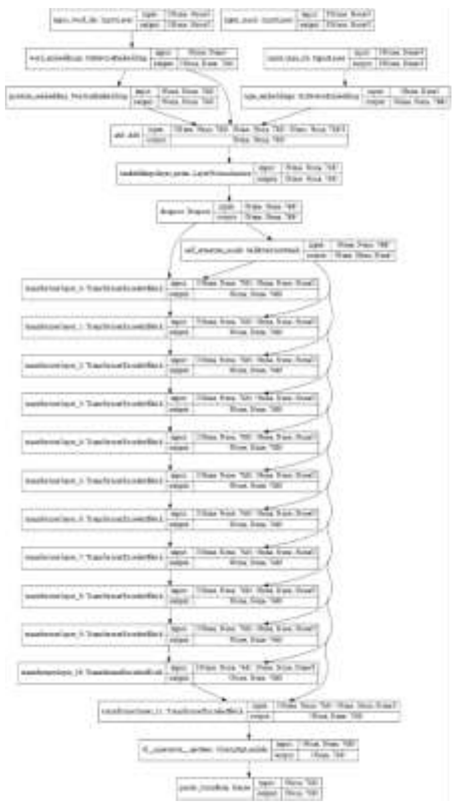
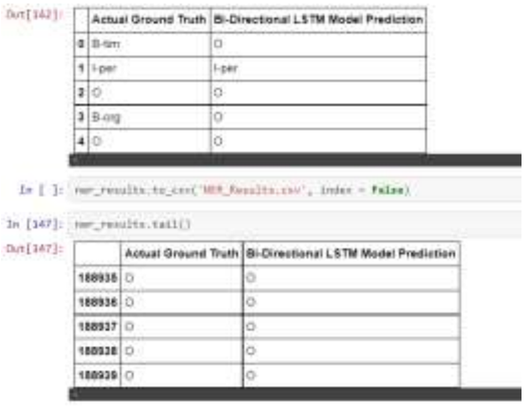


Fig. 23. BERT model



```
Out[142]:
```

	Actual Ground Truth	Bi-Directional LSTM Model Prediction
0	B-tim	O
1	I-per	I-per
2	O	O
3	B-org	O
4	O	O

```
In [ ]: ner_results.to_csv('NER_Results.csv', index = False)

In [147]: ner_results.tail()
Out[147]:
```

	Actual Ground Truth	Bi-Directional LSTM Model Prediction
188935	O	O
188936	O	O
188937	O	O
188938	O	O
188939	O	O

Fig. 23. Testing results

The main part of the whole process was creating the dataset, the model gave good results when all the tags of the dataset were evenly distributed else model gave bias prediction towards the tag which was more in number irrespective of accuracy. The models gave bias prediction when model achieved good prediction and also when model achieved average prediction, so it was found that model gave the wrong prediction that was due to dataset. It was found that creating the dataset was a more causal part followed by building the model

5. CONCLUSION

In this paper, we showed the masking of private data using Named Entity Recognition using Deep Learning concepts and how it can be used for achieving data anonymity. We talked about using various algorithms to achieve this and also portrayed the results achieved. Data is something we should always be very careful with. It describes us, helps us in finding things of similar interest like us but we must remember that it can also be used for manipulating how we think and function. This model recognizes the tags by their value and that value will be masked to generalized detail. In the dataset, we have many tags but we have limited the tags and taken only essential tags according to our application requirements. If there is any person, organization name or so on then the model will recognize it by its tags which will be masked as an output. We hope that our work will enhance the development of such tools soon.

6. REFERENCES

[1] Di Cerbo, Francesco & Trabelsi, Slim. (2018). Towards Personal Data Identification and Anonymization Using Machine Learning Techniques: ADBIS 2018 Short Papers and Workshops, AI*QA, BIGPMED, CSACDB, M2U, BigDataMAPS, ISTREND, DC, Budapest, Hungary, September, 2-5, 2018, Proceedings. 10.1007/978-3-030-00063-9_13.
[2] Vincze, Veronika, and Richárd Farkas. "De-identification in natural language processing." 2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE, 2014.
[3] Medlock, Ben. "An Introduction to NLP-based Textual Anonymisation." LREC. 2006.
[4] Webster, Jonathan J., and Chunyu Kit. "Tokenization as the initial phase in NLP." COLING 1992 Volume 4: The 15th International Conference on Computational Linguistics. 1992.
[5] Manning, Christopher D., et al. "The Stanford CoreNLP natural language processing toolkit." Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations. 2014.
[6] Sherstinsky, Alex. "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network." Physica D: Nonlinear Phenomena 404 (2020): 132306.
[7] Hochreiter, Sepp & Schmidhuber, Jürgen. (1997). Long Short-term Memory. Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735.
[8] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," in IEEE Transactions on Signal Processing, vol. 45, no. 11, pp. 2673-2681, Nov. 1997, doi: 10.1109/78.650093.
[9] Vaswani, Ashish & Shazeer, Noam & Parmar, Niki & Uszkoreit, Jakob & Jones, Llion & Gomez, Aidan & Kaiser, Lukasz & Polosukhin, Illia. (2017). Attention Is All You Need.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” 2018 [Online]. Available: http://arxiv.org/abs/1810.04805

[11] Polikar R. (2012) Ensemble Learning. In: Zhang C., Ma Y. (eds) Ensemble Machine Learning. Springer, Boston, MA. https://doi.org/10.1007/978-1-4419-9326-7_1

[12] L. K. Hansen and P. Salamon, "Neural network ensembles," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 10, pp. 993-1001, Oct. 1990, doi: 10.1109/34.58871.

[13] Y. Liu, X. Yao, Ensemble learning via negative correlation, Neural Networks, Volume 12, Issue 10,1999, Pages 1399-1404, ISSN 0893-6080, https://doi.org/10.1016/S0893-6080(99)00073-8.

[14] MacKay D.J.C. (1995) Developments in Probabilistic Modelling with Neural Networks — Ensemble Learning. In: Kappen B., Gielen S. (eds) Neural Networks: Artificial Intelligence and Industrial Applications. Springer, London. https://doi.org/10.1007/978-1-4471-3087-1_37

[15] David H. Wolpert, Stacked generalization, Neural Networks, Volume 5, Issue 2, 1992, Pages 241-259, ISSN 0893-6080, https://doi.org/10.1016/S0893-6080(05)80023-1.