Identifying and Classifying Personally Identifiable Information

Aditya Peer (2020355)
Ayush Gupta (MT23027)
Sonu Kumar (MT23144)

Abstract

With the rise of remote/online learning, students have to submit their work on various online platforms. These online work resources are usually used as educational datasets for various purposes. However, they also come with a privacy risk. One such risk is the presence of text, which could be used to identify the student. This is known as Personally Identifiable Information(PII). In order to share resources as a public dataset, the dataset providers need to censor/replace PII data from their datasets manually. This is a highly time-consuming and inefficient way to do that. The purpose of this project is to develop a Machine Learning algorithm that can identify and classify PII from a corpus of student essays. In this paper, we discuss our training data, work done till now, and future work to do.

Keywords: Neural Networks, Natural Language Processing, Personally Identifiable Information, Named Entity Recognition, Word Tokenization, Word Embeddings, N-Gram model, RNN, LSTM model

1 Introduction

The problem of detecting and categorizing PII data is of utmost importance. It helps promote privacy and sensitive data protection, which becomes even more important when it involves students. PII data is usually found in textual data. The branch of Machine Learning that deals with working on textual data is known as Natural Language Processing(NLP). Since models cannot read raw textual data, NLP involves certain text preprocessing to use with ML models. One characteristic of textual data is that it is sequential data. A word doesn't exist in a vacuum; its meaning/semantics are highly dependent on the words before and after it, or we can say it is highly dependent on the context.

1.1 Related works

Extraction of PII from unstructured text is a crucial task[1] in many industries. It is a laborious task that can be automated through the use of Named Entity Recognition (NER) models. Rajitha et.al[1] argue that traditional Supervised machine learning methods like Condition Random Field[2] and Support Vector Machines, along with extensive feature extraction techniques, were employed to train NER models in the past; however, contemporary advancements in deep learning, such as Recurrent Neural Networks (RNNs) and LSTM-RNNs, have emerged as preferred choices for NER model training. These models excel in retaining sequence information and preserving context within sentences.

Poornima et al.[3] argue that the identification of Personally Identifiable Information (PII) within unstructured text data is a text mining and clustering challenge. It presents a novel clustering-based model for PII detection (C-PIIM) utilizing natural language processing (NLP) techniques and an unsupervised learning algorithm. NLP methods, including Word-to-Vec and Bag of Word models, are applied for topic modeling. Concurrently, an unsupervised learning algorithm called Byte-mLSTM is employed to identify full PII from the text corpus. Evaluation of the C-PIIM system involves assessing clustering performance metrics and the probability of PII occurrence within the text corpus.

Another model typically used is known as a BERT (Bidirectional Encoder Representations from Transformers) model, which is a transformer-based language model pre-trained on a massive corpus of written text. BERT and a series of language models belonging to BERT's family form the backbone of today's deep learning NLP models.[4]

1.2 Objectives

The objective of our project is to first - identify PII tokens from a corpus of students essays and then second - classify them from various classes. We can do this by developing a ML model which is trained using the corpus of student essays through various techniques.

1.3 Scope and Impact

The scope of this report is to study the dataset, preprocess the data and train some models on the train dataset to learn what challenges we need to overcome. The impact of this is that it will help develop a ML model which can reliably catch and categorize PII data to protect the identity of students.

2 Materials and Methods

2.1 Dataset and Data augmentation

Our training data is a corpus: body of written work, of student essays which comes with an already tokenized list of the essays along with the class they belong to. For our project, we are provided with 2 JSON files: train.json and test.json. Both JSON files contain a list of objects with the following properties:

- **document:** The ID of the document/essay
- **full_text:** The entire raw essay string
- **tokens:** An array containing the essay as tokens
- **trailing_whitespace:** An array of boolean values telling if the corresponding token has a trailing whitespace or not
- **labels(only for train.json):** An array of target labels for the corresponding token

There are 15 target labels tagged in the BIO format: Labels starting with 'B' indicate the beginning of the entity token, Labels starting with 'I' indicate the interior of the entity token, and 'O' labels indicate the non-PII token.

There are 7 PII categories:-

1. **NAME_STUDENT-** Name of the student, which may not necessarily be the author
2. **EMAIL-** Personal email address of student
3. **USERNAME-** A student's username on any platform
4. **ID_NUM-** A number or alpha-numeric sequence of characters that could be used to identify the student, such as a student ID
5. **PHONE_NUM-** A phone number associated with the student
6. **URL_PERSONAL-** A URL that might be used to identify a student

7. **STREET_ADDRESS-** A full or partial street address that is related to the student, such as their home address

The dataset provided only has a small sample of PII data, which makes it hard to train the model. For this reason, we also use a 3rd party dataset from Kaggle which has a comparitively higher number of PII samples as shown in figure:1

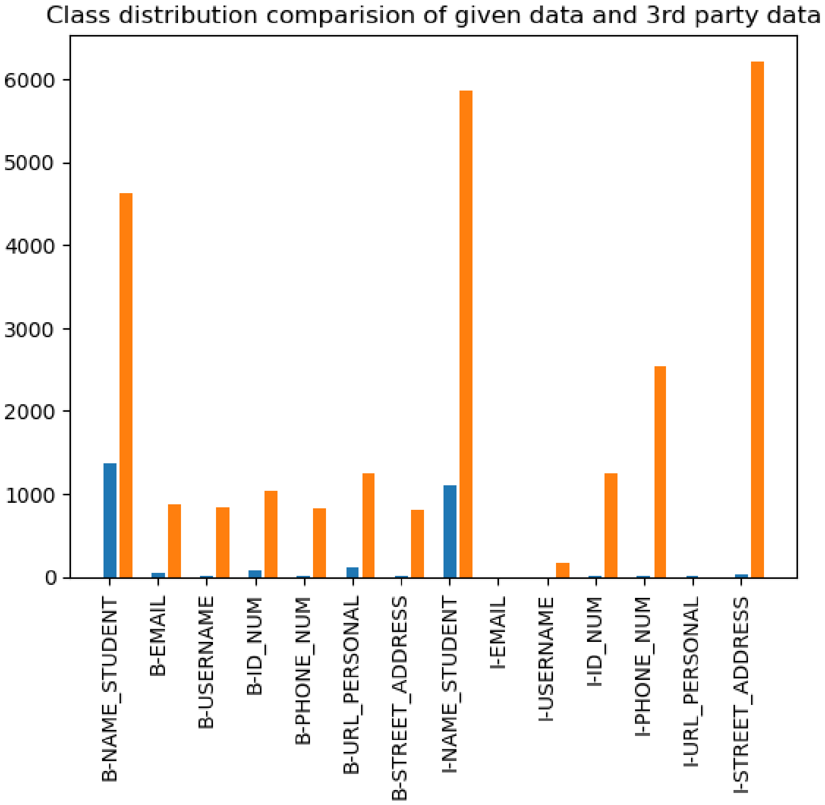


Figure 1: Samples comparision

Also, we see that there is a lot of non-PII data as compared to PII data, which makes our model highly biased towards non-PII data. To combat this, we use downsampling to reduce the number of non-PII tokens to 0.01% of the original number of samples.

Finally, we remove **stopwords** from our training data as they usually don't have a lot of impact on a sentence.

2.2 Methodology

2.2.1 Word Tokenization and Word Embedding

Machine Learning models cannot directly work with textual data. The textual data first needs to be broken down: which is called **tokenization** and then converted into a vector of numbers: which is known as **embedding**.

Tokenization can be done in a variety of ways, however, the most common way is to split a string of words using the 'whitespace' character. It depends on the problem if characters like punctuation or newline needs to be removed during tokenization or not. For our problem, the dataset already comes with the essays in a tokenized form. It contains punctuations and newline characters as separate tokens which is important to our problem. Each unique token in our corpus is part of the **vocabulary** and is assigend a unique id/value.

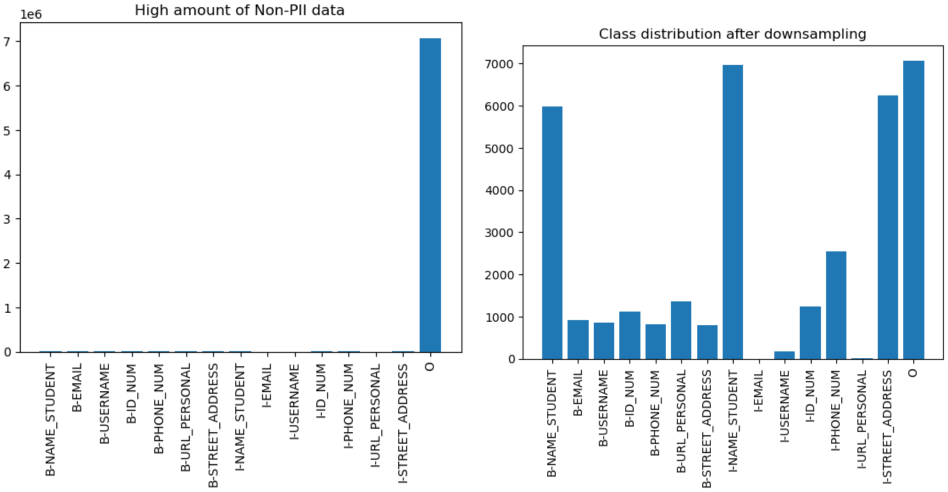


Figure 2: Class comparison before downsampling and after downsampling

Embedding converts a token in the vocabulary of the problem to a vector of a specified dimension. The dimension number is called the embedding dimension. Embedding create a vector which captures different semantic meanings of the words for each component. It could mean anything, it depends upon the ML model to learn those semantic meanings. For our models, embedding is handled by embedding layers of our model.

2.2.2 N-Gram model

The first model we worked on is called an N-Gram model. It is one of the simplest kinds of NLP models which takes in as input a fixed sequence of tokens and outputs the classification logits. The hidden layers are usually fully connected layers. The idea behind an N-Gram model is that since a token depends on the tokens before it, we can send a fixed number of tokens which is called the **context size**.

For our model we used a context size of 10 and an embedding dimension of 10. Hence, the input size of the model is 100. The output size of the model is 15, since we have to classify from 15 classes. There is only one hidden fully connected layer having 128 neurons.

Since we are dealing with multi-class classification, we have used Cross Entropy Loss with softmax as normalization for our loss function.

2.2.3 Pre-Trained LSTM-RNN model

A **Recurrent Neural Network** plays a major role in the NLP domain as it works very well with a sequence of data as input. If we talk of standard Neural Networks in which all of the inputs and outputs are independent of one another, in such cases, we may not be able to work with a sequence of data. In circumstances where we have identify PII, prior words are necessary, and so previous words must be remembered. To handle this task, RNN seems a very good fit; the important part of RNN is the hidden state, which remembers the specific information about a sequence. The information in recurrent neural networks cycles through a loop to the middle hidden layer. However, this usually leads to the problem of vanishing/exploding gradients.

After understanding the role of RNN, we will now understand the **LSTM-RNN model (Long Short-Term Memory Model)**; this model is a specific type of RNN but helps limit the problem of vanishing/exploding gradients. The **LSTM model** is an efficient way to preserve the important context in the sentence. It has 4 important components : (1) Memory Cell, (2) Forget Gate,(3) Input

Gate, (4) Output Gate. Hence, we have used a LSTM model as our second model.Here we also used cross entropy loss with softmax as normalization for our loss function for multi-class classification.

2.2.4 Evaluation Metrics

We have used two evaluation metrics to judge our models and evaluate their performance:-

- **Accuracy:** It helps us in finding the proportion of correct classifications from all the samples. However, it could be highly misleading since our dataset is imbalanced.
- **Confusion matrix:** It gives a clearer picture of what our model is actually predicting and where it is falling short

3 Results and Analysis

3.1 N-Gram Model

After training the model and evaluating it on the test set, we obtained an accuracy of 92.238%. However, after looking at the confusion matrix in figure 3 we see that our model is just predicting a lot of data as non-PII data and since the test set has a lot of non-PII data, it is giving high accuracy.

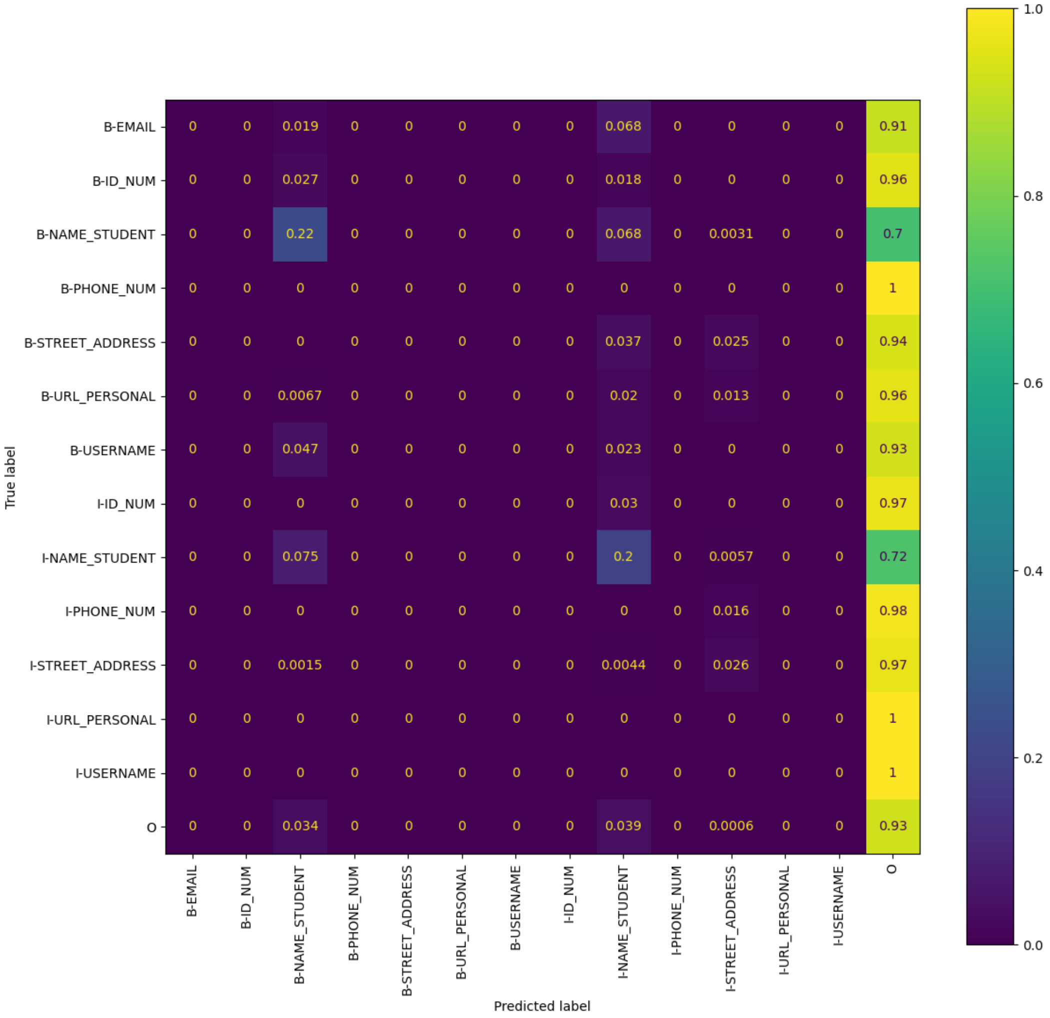
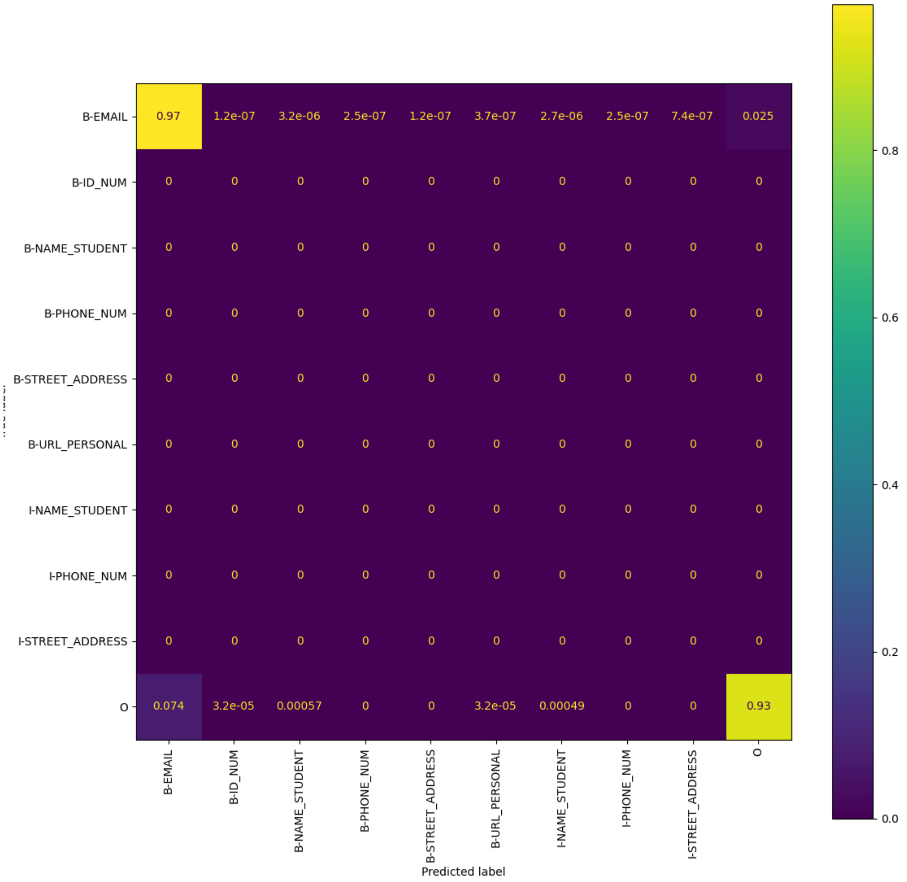


Figure 3: Confusion matrix of the N-Gram model



	B-EMAIL	B-ID_NUM	B-NAME_STUDENT	B-PHONE_NUM	B-STREET_ADDRESS	B-URL_PERSONAL	I-NAME_STUDENT	I-PHONE_NUM	I-STREET_ADDRESS	O
B-EMAIL	0.97	1.2e-07	3.2e-06	2.5e-07	1.2e-07	3.7e-07	2.7e-06	2.5e-07	7.4e-07	0.025
B-ID_NUM	0	0	0	0	0	0	0	0	0	0
B-NAME_STUDENT	0	0	0	0	0	0	0	0	0	0
B-PHONE_NUM	0	0	0	0	0	0	0	0	0	0
B-STREET_ADDRESS	0	0	0	0	0	0	0	0	0	0
B-URL_PERSONAL	0	0	0	0	0	0	0	0	0	0
I-NAME_STUDENT	0	0	0	0	0	0	0	0	0	0
I-PHONE_NUM	0	0	0	0	0	0	0	0	0	0
I-STREET_ADDRESS	0	0	0	0	0	0	0	0	0	0
O	0.074	3.2e-05	0.00057	0	0	3.2e-05	0.00049	0	0	0.93

Figure 4: Confusion matrix of the LSTM model

3.2 LSTM Model

After training the model and evaluating it on the test set, we obtained an accuracy of 97.821% . However, we observe the same as we did in the N-Gram model. A high-class imbalance leads to biased accuracy as we see in figure 4

4 Discussion

From the results, it is clear that we are observing a very high accuracy for the test data set for both models, but looking at the confusion matrix, we see that the model has low sensitivity for all the PII classes. This is because the test set has a lot of non-PII data and our model is currently biased towards predicting a lot of non-PII data even after downsampling.

5 Work to be done

We are trying to implement and use the BERT model which has shown to have very good accuracy for NLP problems. It is evident from our current results that we need to solve the problem of imbalance in our dataset, as even with downsampling, we see that our model is biased towards non-PII data. There are certain solutions that we could employ:

5.1 Weighted classes

We can use a weight for the non-PII classes, which is proportional to its proportion of data. This gives more importance to the minority classes during training, giving better results for them.

5.2 Two step models

We can use two models for our PII classification. Since all PIIs are nouns or noun phrases, we can use Named Entity Recognition(NER) to first filter out all the named entities from our corpus and then use another model to classify the data into our categories. Hence, one model has the job of identifying potential PIIs and the other model to classify it.

6 Work Distribution

1. Aditya Peer- Removal of stop words from the dataset, working on the LSTM model
2. Ayush Gupta- Downsampling of data, working on the N-Gram model
3. Sonu Kumar- EDA of the dataset and working on the BERT model

References

[1] Rajitha , R.H. & Isar, I.N. & Miodrag, M.B. (2020) Privacy-Preserving Approach to Extraction of Personal Information through Automatic Annotation and Federated Learning.*58th Annual Meeting of the Association for Computational Linguistics,*, pp. 36–45.

[2] Gang, G.L. & Xiaojing, X.H. & Chin-Yew, C.L. & Zaiqing, Z.N. (2015) Joint named entity recognition and disambiguation*Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* pp. 879-888

[3] Poornima, P.K. & Cauvery, N.K. (2021) Personally Identifiable Information (PII) Detection in the Unstructured Large Text Corpus using Natural Language Processing and Unsupervised Learning Technique. *International Journal of Advanced Computer Science and Applications* Vol. 12, No. 9,.

[4] Qiang, Q.Y. & Yang, Y.L. & Tianjian, T.C. & Yongxin, Y.T. (2019) Federated machine learning: Concept and applications *ACM Transactions on Intelligent Systems and Technology (TIST)* pp, 1-19