# Personal Identifiable Information Detection in Student Writing
## NLP Project Work

**Pelinsu Acar** and **Călin Diaconu**
Master's Degree in Artificial Intelligence, University of Bologna
{ pelinsu.acar, calin.diaconu }@studio.unibo.it

## Abstract

The project described in the current report represents a solution to the Personally Identifiable Information (PII) Detection public competition, organized on the Kaggle platform. Since the problem can be posed as a sub-type of Named Entity Recognition (NER), traditional NER algorithms are compared with newer developments, namely Large Language Model (LLM) based classificators. After alterations of the data and of the basic models are tested, such as extending the training set with automatically generated entries, and using hybrid models, where regex based rules are employed, the vanilla version of a traditional NER model, namely the spaCy one, turns out as the leader among the other solutions. It is not the best one on the public validation set, but it gives an F5 score of 0.743 on the hidden test set, outperforming the other setups with at least 5%.

## 1 Introduction

The current project has been develop with the aim to take part in the Kaggle competition (Langdon Holmes, 2024), organized by the Vanderbilt University from Nashville, Tennessee, in partnership with The Learning Agency Lab, that gives out the task of Personally Identifiable Information (PII) data detection.

PII represents any data that could be used to identify a specific person, by distinguishing individuals, and thus deanonymize previously anonymous data.

The goal of the competition is to obtain a PII detection solution that could be applied on student essays, such that, in the future, the labelling of this type of datasets will become less expensive and time consuming. In turn, according to the organizers, this should support the development of educational tools by enabling the usage of the largely available educational data in automatic learning contexts, after cleansing it of any sensitive information.

To accomplish this, 4 solutions were developed, ranging from traditional methods, to more novel LLMs, with alterations on these, such as including automatically generated data, yielding inferior results compared to the default models.

## 2 Background

The most reliable PII screening method available at the current time is the manual review of the entire dataset, with the obvious drawback of cost, which restricts the scalability of educational datasets.

However, a number of automatic solutions also exist, relying on named entity recognition (NER). They work best with data that shares a common format, such as telephone numbers, email addresses, and social security numbers (SSNs), some of them being based on regex rules. Oftentimes, they struggle to correctly label names, and especially to differentiate between sensitive names (such as the name of the student writing the essay), and non-sensitive ones (such as the name of a cited author).

Most of the existent solutions are already integrated in popular software libraries, such as traditional and transformer-based (specifically, RoBERTa (Liu et al., 2019)) models available with SpaCy (Honnibal and Montani, 2017), the named entity chunker of the NLTK library (Bird et al., 2009), Microsoft's Presidio SDK (Mendels et al., 2018), and more modern LLM-based solutions, that fine-tune pre-trained models for the task of NER (Tjong Kim Sang and De Meulder, 2003) (Devlin et al., 2018), most of them available through the HuggingFace platform (Wolf et al., 2020).

## 3 System description

The project consists in exploring a number of models, with the purpose of finding the best performing one. These models can be split into 2 groups: the first one is based on traditional NLP methods, which are used as baselines, while the second group involves large language models.

### 3.1 Baseline Models

We used two open-source libraries for python that don't require actual training to define a baseline for our large transformer-based models. One of them is NLTK and the other is spaCy.

#### 3.1.1 NLTK

NLTK (Natural Language Toolkit) is a popular library for natural language processing tasks in Python. While it offers a wide range of functionalities, we specifically used for named entity recognition (NER) to detect student names, which are a subset of PII.

NLTK provides pre-trained models for NER, including one for recognizing person names. Among the entities NLTK supports, only the detection of 'STUDENT_NAME' is possible. However, considering that 'STUDENT_NAME' is the prevalent class (see section 4), we have concluded that the performance will provide a valuable insight as a baseline.

The input text is tokenized with NLTK tokenizer and passed through the NLTK NER model. Performing a POS tagging for each chunk, we identified the entities labeled as 'PERSON'.

#### 3.1.2 spaCy

spaCy is another popular Python library for NLP tasks, offering functionalities for tokenization, POS tagging, dependency parsing, and NER, among others. Similar to NLTK, spaCy provides pre-trained models for NER, including one trained on the large English language corpus.

We specifically utilized the 'en_core_web_lg' model, which is a English pipeline optimized for CPU and had several components like tok2vec, tagger, parser, senter, ner, attribute_ruler, lemmatizer. However we disabled all unused components other than ner.

The input text is processed through the spaCy NER pipeline. Unlike NLTK, spaCy supports 'CARDINAL' entity which can be used to detect phone and ID numbers. So we applied some rule-based filters for entities detected as 'CARDINAL' to categorize them into phone and ID numbers. Then all the predicted entities, including person names are compared against the ground truth annotations.

Similar to the transformer-based models, we evaluated both NLTK's and spaCy's performance using standard NER metrics such as precision, recall, and F5-score which is specified by the leaderboard of the competition (see section 5.1).

### 3.2 Transformer Based Models

We tried 2 different LLM models which are Deberta Base and Distilbert Base. For a fair comparision, a common training recipe and custom tokenization function are used.

We defined a custom tokenization function to tokenize the input text data for training. This function takes an example from the dataset, along with a tokenizer, label-to-id mapping, and maximum length parameter. It processes the input text by iterating through the tokens, labels, and trailing whitespaces. It appends the tokens to a list (text) and extends the labels for each token. If there's trailing whitespace, it appends a space and a corresponding "O" label to represent non-entity tokens. Finally, it maps the labels to token indices based on the offset mapping obtained from tokenization and returns a dictionary containing the tokenized inputs, token-level labels, and length of the tokenized sequence.

#### 3.2.1 Deberta Base

DeBERTa, short for Decoding-enhanced BERT with disentangled attention, is an advanced transformer-based model for natural language processing (NLP) tasks. It is an extension of BERT (Bidirectional Encoder Representations from Transformers) with several enhancements designed to improve its performance on various NLP tasks. It focuses mainly on positional encoding. It utilizes a disentangled self-attention mechanism, which separates content-based attention from position-based attention with an enhanced mask decoder, in which both relative and absolute position of words are taken into account.

Unlike in traditional transformer-based models like BERT, DeBERTa includes additional embeddings specifically dedicated to encoding the absolute positions of words within the input sequence, independent of the relative positions learned through the transformer layers, as seen in Figure 1. These absolute word position embeddings provide fixed positional information for each word in the sequence and are not updated during training by incorporating absolute word position embeddings separately to have more explicit and fine-grained control over the absolute positional information of words in the input sequence.

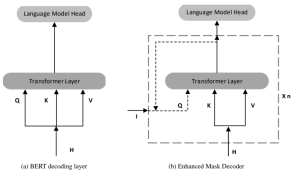


Figure 1: The comparison of Bert and Deberta decoding layers

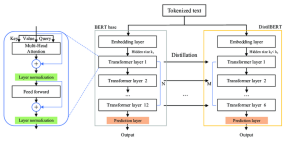


Figure 2: The DistilBERT model architecture and components

### 3.2.2 Distilbert Base

While other models are designed to improve the performance of BERT, DistilBERT has a different objective. Its primary aim is to reduce the large size and enhance the speed of the BERT model while preserving as much of its performance as possible.

DistilBERT achieves this objective by significantly reducing the number of parameters in the model. The original BERT model has 110 million and 340 million parameters (referring to BERT-base and BERT-large, respectively), whereas DistilBERT substantially reduces this size. Additionally, DistilBERT improves the speed of inference by a significant margin compared to BERT.

DistilBERT maintains a similar general architecture to BERT, including the use of transformer-based encoder blocks. However, it reduces the number of encoder blocks from 12 in BERT-base and 24 in BERT-large to only 6 blocks in DistilBERT, as seen in Figure 2. Additionally, it removes certain components such as token-type embeddings and the pooler, further reducing the model's complexity and size.

## 4 Data

In this section, the dataset's description, structure, and origin will be analyzed.

### 4.1 Dataset Details

The data associated with the competition comes from a massively online open course (MOOC) (Liedtka, 2017), that teaches tools that facilitate the design of innovative solutions for complex problem solving, with the essays probably originating from its fifth module that has the following prompt:

> The assignment requires selecting one of the design thinking tools presented in the course, writing a reflection, and completing three peer reviews.

In terms of dataset specificities, as a conclusion of the essays' origin, they will almost always refer to one tool in particular, and they share a common format. As a reflection is asked, the essay can be written in first person and contain personal experience information.

Both a training, and a testing set are provided. The former contains 6,807 samples, with the latter being just a demonstrative test set, containing a subset of 10 of the training set samples, but without the labels. The actual test set used for evaluation is hidden for all users, both in terms of the labels, but also in terms of the actual input, and is being used in an automatically run notebook on the competition's platform. This hidden set represents about 70% of the total data available to the organizers, which in turn amounts to about 22,000 essays.

For each training sample, the information provided is an integer, representing the document ID, the text of the essay as a string, a list of string tokens, obtained using the spaCy tokenizer (Honnibal and Montani, 2017), a list of boolean flags, marking whether the token contains trailing whitespaces, and a list of labels.

They are provided in the BIO format, described in (Ramshaw and Marcus, 1995), with 7 classes to identify: the student-author's name, their email address, their username on any platform, an ID or SSN number, that could be used to identify the student, their phone number, URLs that might be used to identify the author, and full or partial geographical addresses associated with them.

### 4.2 Data Analysis and Statistics

Regarding the token-document distribution, three points could of interest. The first one represents the number of tokens per essay, with a peak around the 750-800 area, and a tail of exceptions that stretches beyond the 3,000 mark, as seen in Figure 3. This should give a good image of the essays' lengths, but is not representative of the class-label distribution, since only 13.9% of the essays in the training set contain any PII tokens. Related to this, a third aspect can be observed, namely the distribution of PII tokens inside the essay, represented in Figure 4, giving an idea about the usual position of these elements in the text.

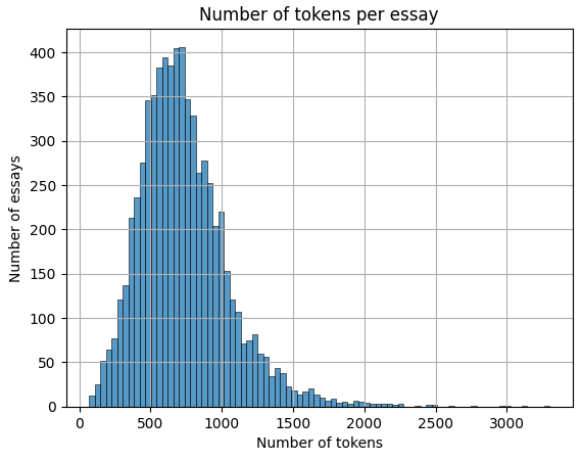


Figure 3: Per-essay Token Distribution

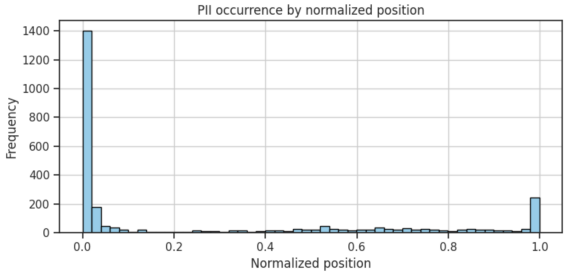


Figure 4: Intra-essay PII Token Occurrence by Normalized Position

On the topic of PII tokens, beside the fact that they are severely outnumbered by non-PII (O) tokens (4,989,794 to 2,739), a strong class imbalance is also present. The NAME_STUDENT class is prevalent, with 1,365 B tokens, and 1,096 I tokens. The next most numerous class is URL_PERSONAL, with a set that is 22 times smaller (110 B tokens, and a single I token). There are 2 classes with 0 labels, and 7 with less than 10 labels. Their distribution, excluding the NAME_STUDENT class for a more clear illustration, is shown in Figure 5. In total, there are 1,601 B tokens, and 1,179 I tokens.

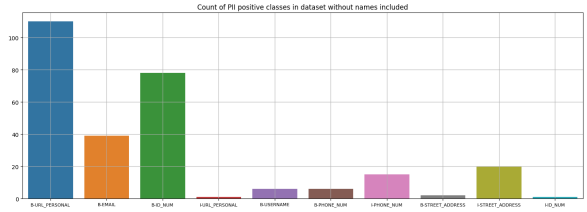


Figure 5: Class Distribution, Excluding the NAME_STUDENT Class

## 5 Experimental setup and results

Here, the metric computation is explained, as well as the evaluation setup.

### 5.1 The F5 Score

The leaderboard of the competition is decided according to a custom metric derived from the $F_\beta$ score, defined in (Rijsbergen, 1979) as:

$$F_\beta = (1 + \beta^2) * \frac{precision * recall}{(\beta^2 * precision) + recall}$$

Thus, the recall will be $\beta$ times as important as the precision. The value for $\beta$ chosen by the competition organizers is 5, so the name used for the metric is the F5 score.

Moreover, it is worth mentioning that the final F5 score is computed using the micro-average, such that the number of labels of each class is taken into account when weighting the result, rather than simply averaging in an equal measure over all classes.

This is the score that will be reported in the current section, as computed by the official notebook of the competition. However, this number is obtained from only 25% of the hidden test data, with the rest of it being used to calculate a private result, that will be published after the final deadline of the competition.

### 5.2 Hardware and Software Setup

The notebooks in which the solution is developed are run on the public Kaggle platform, which offers Intel Xeon CPUs, and NVIDIA Tesla P100 GPUs. There is no limit on the training time (with the exception of the competition running time, between January 17, and April 24 2024), as well as on the resources used for this (the participants are allowed to use both pre-trained models, as well as datasets, that must be freely and publicly available). However, the difficulties arise when evaluating the model on the private test set. While access to a previously trained model that is stored locally is allowed, the evaluation notebook has a running time limit of 9 hours, and it must be disconnected from the Internet, thus limiting the user from using external processing, such as cloud solutions from Google, or chat bots, like the popular Chat GPT.

A secondary competition, titled by the organizers "The Efficiency Prize", has the aim of obtaining lightweight models that still give a good performance, since modern, well performing LLMs

| | NLTK | spaCy | DeBERTa | DistilBERT |
|---|---|---|---|---|
| **F5 Score** | 0.365 | 0.743 | 0.693 | 0.648 |

Table 1: The F5 score of each model, as reported on ∼25% of the hidden test data, by the official evaluation notebook of the competition

are usually very computationally expensive. For this track, an extra limitation is imposed, where a qualifying notebook can only be run on the CPU, without any GPU acceleration, with all other rules previously mentioned still in place. A custom metric is used, called the "Efficiency score". It is based on the F5 score, but it offers a bonus that is inverse proportional to the execution time of the notebook. It is computed as follows, where Benchmark is the score of a provided benchmark, max$F_5$ is the maximum $F_5$ score obtained by a participant in the private leaderboard, the RuntimeSeconds represents the running time of the notebook that is tested, and 32,400 is the number of seconds in 9 hours:

$$Efficiency = \frac{F_5}{Benchmark - maxF_5} + \\ + \frac{RuntimeSeconds}{32,400}$$

## 6 Discussion

In this section, the results reported in Table 1 will be evaluated, and a number of possible future improvements will also be mentioned.

### 6.1 Results Observations

While on the validation set the models stand differently, with the LLMs obtaining clearly superior results, on the hidden set the spaCy model outperforms the others, by a significant margin (5% from the second-best performing model, DeBERTa). Traditional models may be better because of the relatively static format of the prevalent category, the student names, since personal names are one of the most popular classes for NER solutions. Another reason for the overfit of the LLMs could be the relatively small training set. Extending it with automatically generated data didn't give out better results, aggravating the overfit, but some fine tuning of this set generation may be explored in the future, such that the obtained set would be better generalized.

Regarding the efficiency track, the spaCy model was submitted, since it's not run on the GPU (in practice, the time between a CPU and a GPU run vary by a very small value), and it also happens to be the best performing solution.

### 6.2 Future Improvements

A number of other methods were tested, such as included essays in the training set that were generated by other LLMs, and mixing hard-coded regex-based rules with the previously mentioned models. However, all these were excluded from the current report, as they yielded inferior results.

Other alterations that may improve the proposed solutions, but that were not tested during the development of the current project, could include a modified loss, that takes into consideration the increase penalization on recall, given by the F5 score, a GAN-style pair of classificator - essay generator, where both models are optimized, such that higher F5 scores are obtained, better-suited LLMs and generated datasets, or other types of hybrid solutions, or 2-stage and ensemble models, where they are selected based on the type of label on which they give the best results.

For the "efficiency" track of the competition, where the models must be run on the CPU, both lighter LLMs, as well as traditional methods of slimming down models, such as quantization or pruning, could be employed.

## 7 Conclusion

The project described in the current report started as a solution to a public competition that had as task an NER type of token classification, targeted at identifying private information in scholarly essays. After analyzing a number of models and alterations of them, the best performing one was an adaptation of a vanilla spaCy model that transformed the original classes into the required ones. While it was not the best model on the validation set, it managed to obtain an F5 score of 0.743 on the hidden test set provided by the organizers.

## 8 Links to external resources

The official page of the Kaggle competition is this one: https://bit.ly/3TSxdOJ. The notebook can be found on Google's Colab platform, at this location: https://bit.ly/3U9jB2Z. The models are saved on Google Drive, here: https://bit.ly/3Ji7Zo1.

## References


Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Perpetual Baffour Jules King Lauryn Burleigh Maggie Demkin Ryan Holbrook Walter Reade Addison Howard Langdon Holmes, Scott Crossley. 2024. The learning agency lab - pii data detection.

Jeanne Liedtka. 2017. Design thinking for innovation.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Omri Mendels, Coby Peled, Nava Vaisman Levy, Tomer Rosenthal, Limor Lahiani, et al. 2018. Microsoft Presidio: Context aware, pluggable and customizable pii anonymization service for text and images.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. *CoRR*, cmp-lg/9505040.

C. J. Van Rijsbergen. 1979. *Information Retrieval*. Butterworth-Heinemann.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.