BUILDING A PERSONALLY IDENTIFIABLE INFORMATION RECOGNIZER IN A PRIVACY PRESERVED MANNER USING AUTOMATED ANNOTATION AND FEDERATED LEARNING

Rajitha Hathurusinghe

A thesis submitted in partial fulfillment of the requirements for the
Master of Applied Science in Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Acknowledgments

First and foremost, I would like to thank my supervisors Miodrag Bolic and Isar Nejadgholi who have been crucial in defining the research problem and directing me in this research topic. I thank Dr. Isar for guiding me through the latest trends in privacy research and deep learning based natural language processing research. And professor Bolic for guiding me through research practices consistently and teaching me to ask important research questions.

I thank my colleagues Shan He, Zixiong Han, Fan Yang, Varun Mehta and Cristóvão Iglesias at Computational analysis and acceleration research group (CARG) for working together in solving various interesting research problems and their ideas in developing my abilities.

I thank Isuru Gunesekara for introducing me to professor Miodrag and IMRSV Data Labs. I like to thank everybody at IMRSV Data Labs where I could spend the rest of my time doing my research work. I like to remind Sam, Benette, Henri, Alex, Qianhui, Hichem, Serena, Aleksander, Jason, Melissa, Harrison and Abhijeeth. Discussions and knowledge sharing sessions with you all have been crucial in my thinking while proceeding with the research work. I am also thankful for the friendly and motivating atmosphere at the IMRSV Data Labs.

I thank IMRSV Data Labs and Mitacs for generously funding my research through Mitacs Accelerated. I am beyond grateful in receiving such tremendous support for my research.

I also thank all the friends I met for the first time in Ottawa including many friendly Sri Lankans who made me feel like back home here and being supportive in many ways.

Finally I would like to thank my parents for their support throughout my education at University of Ottawa.

Table of Contents

List of Tables

List of Figures

Acronyms

BERT Bidirectional Encoder Representations from Transformers. 6, 21

DP Differential privacy. 12

FL Federated Learning. 4–6, 9, 23, 24

GDPR General Data Protection Regulation. 12

HIPAA Health Insurance Portability and Accountability Act. 11, 12

LSTM Long Short Term Memory. 16, 17

NER Named Entity Recognition. ix, 2, 5, 6, 8, 9, 15, 17, 23, 49

NLP Natural language processing. ix

PII Personally Identifiable Information. iii, 4–7, 9–13, 55

SVM Support Vector Machine. 14

Abstract

This thesis explores the training of a deep neural network based named entity recognizer in an end-to-end privacy preserved setting where dataset creation and model training happen in an environment with minimal manual interventions. With the improvement of accuracy in Deep Learning Models for practical tasks, a rising concern is satisfying the demand for training data for these models amidst the concerns on the data privacy. Several scenarios of data protection are suggested in the recent past due to public concerns hence the legal guidelines to enforce them. A promising new development is the decentralized model training on isolated datasets, which eliminates the compromises of privacy upon providing data to a centralized entity. However, in this federated setting curating the data source is still a privacy risk mostly in unstructured data sources such as text.

We explore the feasibility of automatic dataset annotation for a Named Entity Recognition (NER) task and training a deep learning model with it in two federated learning settings. We explore the feasibility of utilizing a dataset created in this manner for fine-tuning a state-of-the-art deep learning language model for the downstream task of named entity recognition. We also explore this novel setting of deep learning NLP model and federated learning for its deviation from the classical centralized setting.

We created an automatically annotated dataset containing around 80,000 sentences, a manual human annotated test set and tools to extend the dataset with more manual annotations. We observed the noise from automated annotation can be overcome to a level by increasing the dataset size. We also contributed to the federated learning framework with state-of-the-art NLP model developments. Overall, our NER model achieved around 0.80 F1-score for recognition of entities in sentences.

Chapter 1

Introduction

Due to the availability of hardware resources and advancements in training, deep neural networks have improved to solve real world tasks efficiently. These models rely on a vast number of data points to achieve accurate predictions on unseen data. However, most industries and applications beneficial for the day-to-day life have a limitation in datasets. Therefore, sharing data to a common operator by distributed organizations can contribute in creating rich datasets. Unfortunately, this is not possible due to not having consent from data owners or other unavoidable ethical conflicts, all of which boil down to the privacy of the data source. Meanwhile, laws have been strengthen due to the heavy exploitation of privacy sensitive data without privacy concerns in the recent past. General Data Protection Regulation (GDPR) [1] imposed by European Union which is also adopted by North America in the recent past is a major law governing the data privacy. This law gives the data owning individuals the right to demand the data associated to them to be forgotten, which is a basic concern of every individual. With this clear statement on data removal, a collecting organization cannot share these data to a third party to exploit it for creating a dataset. These concerns of guaranteeing the data removal, along with the possible security breaches related to data sharing, impose the requirement for isolation of data within the collecting organization. Federation of datasets and models can be utilized to give strong guarantees of privacy along with concordance to the mentioned civil laws.

Traditionally, NLP tasks demand various language resources for lexical acquisitions which are useful in learning the syntactic and semantic nature of the tasks [2]. Similarly, underlying concepts of machine word representations such as word embeddings in NLP, also used to depend on the task specific datasets. But relying on a specific dataset for generalising the semantics of language in deep learning has failed over the time [3]. With the availability of better hardware resources and robust training algorithms, latest deep learning NLP models can model the language with many diverse sources of datasets [3]. Latest architectures such

as GPT-2 have given exceedingly optimistic results on traditional language model tasks with the underlying guarantee that the system has higher capacity to learn from a vast number of examples to perform the task. However, utilization of these models in realizing practical tasks is limited by the availability of diverse data points describing the task for the model to train on.

Privacy and confidentiality are the biggest barriers in utilizing many diverse occurrences of language demonstration in creating datasets to learn useful tasks in the fields of medicine, law, defense, media and retail. In crucial fields, such as medical text processing there are shared task initiatives, i2b2[1] (Informatics for Integrating Biology to the Bedside) challenge for example. These efforts scale up slowly to create general datasets for natural language understanding amidst the issues of privacy[4].

Automated annotation and utilizing data with federated learning for model training can preserve privacy of sensitive data to a greater extent by leaving data in isolation. Isolation of data sources while curating them with automated annotation is intuitively a less risky approach in utilizing sensitive data to create larger datasets where manual annotation is impossible due to privacy demands. But total isolation of data sources with automated annotation is bound to introduce more noise to the dataset than manual annotation. Deep Learning models have been found to be resilient to the noise in labelling with increased amounts of training data [5]. But models should have the capacity to learn from a larger amount of data points to benefit from a privacy preserving method which introduces noise to the training data. It is also suggested that most learning based NLP algorithms are not robust to gain performance enhancements with the increase of dataset size in NLP tasks such as information retrieval [6]. With the new state-of-the-art NLP models' capacity to be trained on massive datasets [7], with reduced risk of over-fitting due to pre-training [8] and with minimized training bottlenecks of earlier models, learning with large amounts of noisy data is a possibility to be explored. This can al be investigated in achieving the privacy guarantees given by the automated annotation.

To find the solutions to the above research problems, we programmatically created a dataset and trained an NER using federated learning. We created a dataset with common personally identifiable information as extracted entities. The dataset was created with an automated pipeline. Then we trained an NER to recognize entities in unstructured text data in a scenario where data is assumed to be located remotely. We present the test results of the trained model to evaluate the performance guaranteed by this end-to-end privacy preserving approach. We further evaluate the impact on accuracy upon utilizing federated learning for dataset isolation.

[1] https://www.i2b2.org/

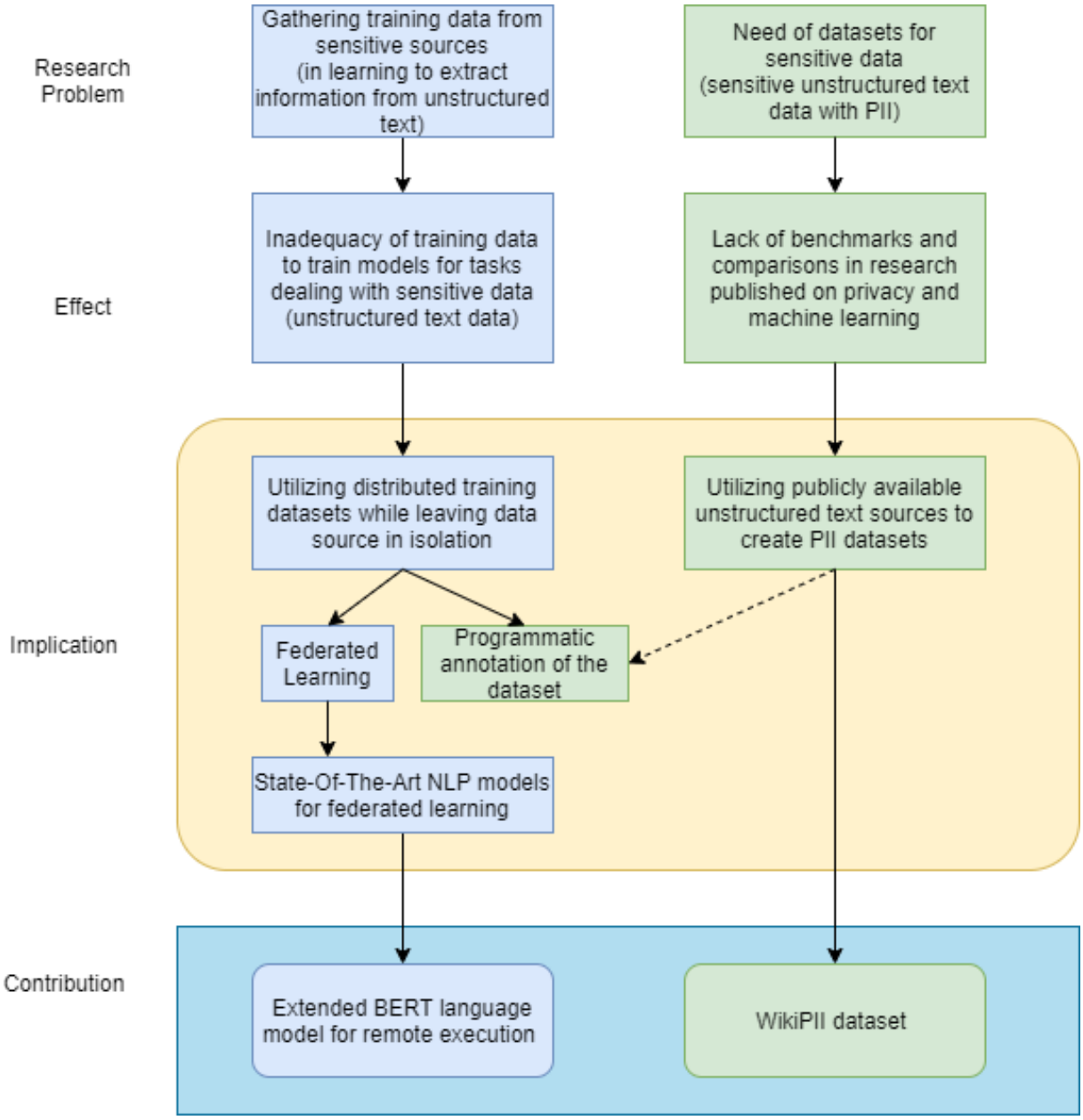

Figure 1.1: An overview of the research problems associated with the thesis topic.

An overview of the problems associated with the thesis are given by the Figure 1.1. We identify gathering trainable data from sensitive sources as a problem which causes the inadequacies of training data. It implies the automated annotation and data source isolation as solutions to address it. We contribute by designing experiments to create a dataset programmatically and utilizing it for training in Federated Learning (FL) setting.

In our experiments, we trained a NER to recognize entities in unstructured text data in a scenario where data is assumed to be located remotely. We present the test results of the trained model to evaluate the performance guaranteed by this end-to-end privacy preserving approach. We further evaluate the impact on accuracy upon utilizing federated learning for dataset isolation.

Due to the novelty of privacy research in the fields of deep learning and NLP, it is quite challenging to publish research work without proper datasets. This affects the advancement of privacy based research due to the lack of benchmarks and comparisons of methods on suitable datasets. We contribute a dataset containing Personally Identifiable Information (PII) for this domain of research created by utilizing Wikipedia biography pages.

1.1 Motivation

Extraction of desired information from unstructured text data is a crucial task in many industries. Extraction tasks are basically identifying named entities. This extracted information is utilized for indexing documents, adding links, categorizing, etc. Automation of such laborious text manipulation tasks saves time and cost, which is a primary concern in various fields such as medicine [9]. These tasks have been done by deep learning models recently with improved accuracy [10].

Accurate recognition of information from unstructured text data is a crucial task. But training a model with text data from a single location would restrict a model. Ideally, a conglomerate of all the data gathered on a certain subject by several locations would manifest a rather complete picture of the problem. But with the concerns of data privacy this is not a possibility. When curating data to one training set, both centrally gathering data and annotating it by human effort increase the risk of violating the privacy. Even if these concerns are addressed by de-anonymization or by adding noise (given the data holders have adequate consent) to the released data, there are possibilities for privacy violations. A famous case of such privacy violation due to anonymous public datasets has been studied in [11], where anonymous preferences were matched to a named database hence exposing individuals. Due to these well-known concerns on data breaches laws to govern such situations, such as GDPR, introduce two articles where the right to withdraw consent and the right to

be forgotten are granted. These laws strictly forbid a data holder from using the consent given by the individuals in releasing the data to the public as a dataset. Pre-existing privacy guaranteeing mechanisms which are blurring the lines of these strict laws are becoming obsolete. Such concerns on data privacy imply the solutions should be focused more on the data source isolation as a privacy preservation method when compared with the existing mechanisms. Improvement of performance achieved in learning a PII recognition task by utilizing a data source while leaving it in isolation contributes towards preservation of privacy. We experiment with our combination of automated annotation, federated learning and the usage of a state-of-the-art transformer based NER model in achieving this privacy goal.

With motivation to find a solution to preserving privacy while utilizing sensitive data, we explore the possibility of totally isolating data to preserve privacy and using these data sources to remotely train a deep learning model in achieving a NER task. Here, the concept of distributed isolated datasets has to be realized in the training paradigm. We use a novel framework based on Federated Learning (FL) to build this mechanism. We modify and train a state-of-the-art pre-trained language model with federated learning which is an understudied possibility that is crucial and novel in guaranteeing the privacy of unstructured text data. We further explore the effect of federated learning compared to the centralized training setting.

Utilization of the trained model is the final beneficial outcome and is made available to the public while data owners have total control and confidentiality of their data. Exploring the feasibility of a method such as federated learning which can realize isolation-based privacy was our overall goal.

Privacy research in deep learning is novel and driven to explore ways to utilize sensitive data in training machine learning models. Datasets are a vital need in the domain of deep learning. Most research done on privacy based tasks are done in isolation due to the sensitive nature of the utilized data. Research on privacy guaranteeing mechanisms such as de-anonymisation published on a common dataset can provide general baselines and comparable results to drive towards improvements. We choose a source that can potentially be automatically annotated containing several common Personally Identifiable Information (PII) and of massive scale in contrast to the other datasets which are more domain focused and smaller in size.

Due to most popular research in federated learning and deep learning being highly experimental and proofs of concepts with fundamental models and datasets, we find the necessity of exploring a practical use case of state-of-the-art language models in this domain. While following our goal of privacy by isolation, our motivation is put into action by utilizing a federated learning platform (PySyft [12]) to fine-tune a pre-trained state of the art language

model, BERT [13]. Currently, this framework only utilizes federated learning for training basic model structures. Implementing FL for a complex structure under this framework is one of our contributions.

1.2 Summary of Contributions

For our objective of achieving privacy by isolation of the data source, we utilize a state-of-the-art pre-trained language model and fine-tune the model with federated learning for the task of NER. We contribute a modified model for remote execution and and show the impact of our modification compared to the typical training scenario. We use automated annotation in creating a dataset for this setting of isolated data sources. We present the results for the combination of automated annotation and federated learning. We investigate if the dataset size can overcome the compromise of the annotation quality. We provide a large dataset with PII which contains a manually annotated test set and also tools to improve the annotations.

Programmatically Annotated WikiPII Dataset

The created dataset was programmaticly annotated with Personally Identifiable Information (PII) as named entities from Wikipedia biography pages. We present the details about the data source and created dataset in Section 3.1. We present the method of implementing the entity tagging pipeline for curating the data source under Section 3.1.2.

Centralized Training

We train the *BERT-base* NER model on our dataset and observe the performance on manually annotated test dataset and the programmatic test set, given in the Section 3.2.

Federated Model

Our extensions of the model for remote execution which are essential in a federated learning setting are presented in Section 3.4.

Federated Learning

We contribute to the domain of federated learning by exploring the impact of distributing training dataset on the accuracy. We carry out different training scenarios for this. For more

accurate insights and for further investigations, we also train the model with CoNLL-2003 NER dataset. These experiments are listed in the Section 3.5.

Results and Discussion

In Section 4.1, we present the results and discussion for our dataset creation. Detailed statistics on dataset are presented in this section. Also, we present comparison of automated annotations vs. the manual annotations in order to investigate the feasibility of utilising this sort of a tagging process for automated annotation in comparisons to the yielded test results later.

We present the training results in Section 4.2 for our insights to a real world scenario of total isolation of a data source and automated curating when used in a NER task by training a deep neural network. Section 4.2.2 presents the performance comparisons of central vs. federated scenario, and it also shows results related to convergence of loss and test accuracy.

1.3 Boundaries and Limitations

Our training experiments are carried out for one epoch of training due to the availability of resources and to make final results comparable. We also impose this limitation due to the pre-trained *BERT-base* model's ability to reach more than 0.9 F1-score on popular NER tasks upon fine-tuning for one epoch which guarantees an adequate number of training iterations. In creating the dataset programmatically, we utilize the info box as the only input to extract the PII belonging to the biographical entry. We believe this choice will give an insight if such an automated annotation were to be carried out on a sensitive source with limited access. To keep the overall automated annotation and choice of a high capacity model like *BERT-base* accountable for the final results, we also avoid using any weak supervisions such as alternate tag suggestions from models or other instances of tagging for raw data from Wikipedia in creating the dataset.

1.4 Organisation of the Thesis

In Chapter 2, we review the relevant background literature on the problem. Section 2.1 details the laws that are changing the requirements of methods guaranteeing the data privacy. Section 2.2 describes Personally Identifiable Information (PII) which is the form of private data we focus in the thesis. Section 2.3 gives a brief literature on automated annotation of

text data. Named Entity Recognition (NER) is explained in details in the Section 2.4 as it is the task solved under this privacy guaranteeing mechanism.

Chapter 3 details the methods involved in achieving the contributions. Next, we present results from our experiments in Chapter 4. These results closely follow the experiments mentioned in the methodology. We provide the discussion of results under each subsection in the same chapter. Chapter 5 presents our conclusion, summary of contributions and future work.

Chapter 2

Background Material

This chapter explains the background for our research, which spans over privacy, named entity recognition and federated learning. We present the literature on privacy under Section 2.1. We choose the problem of recognition of Personally Identifiable Information (PII) data as our model's task to learn, so we present the importance of PII in Section 2.2. We created our PII dataset with automated entity tagging in unstructured text, so we present background on automated text dataset annotation in Section 2.3. To understand the overall results of our primary experiment, background on Named Entity Recognition (NER) is crucial. We present the explanation of the task and its evaluation metrics in details in the Section 2.4. Learning of the NER task was carried on a state-of-the-art pre-trained language model *BERT-base*, we introduce the model and its internal structure in a simplified manner in Section 2.5.1. We utilized Federated Learning (FL) as our data protection method, we present the concept, application in popular systems and the framework we utilized for our experiments in Section 2.6

2.1 Privacy

Privacy is the control over the personal information by an individual. With recent laws governing the privacy due to the need of general public utilization of private data, guaranteeing privacy has become a challenge in areas of machine learning and data science. Privacy is demanded by individuals for perfectly good reasons while private data is demanded by processing parties for equally beneficial outcomes. Demand of privacy leads to the responsibility of the collected organizations in guaranteeing the confidentiality of the private data. This responsibility has been guarded by the laws. Confidentiality refers to the duty of the collectors entrusted with data to keep that information private. This section explains the laws and the impact on utilizing private data while being compatible with these laws.

Privacy and personal data are a bigger concern than ever with growing technology surrounding exploiting them for our own benefit, yet the boundaries have to be maintained to preserve the fundamental meaning of privacy. Privacy is also generally referred to as the protection given to an individual's personal space. An individual's personal space is defined by personal data for the most part, so any processing that involves personal data can violate the personal space. Privacy can be seen in several ways fitting the information age as well as traditionally, it can be the right to be left alone [14], or as control over the shared personal information hence to have the freedom of judgment from the society [15]. Privacy is spanning across dimensions that are competing and contradictory due to the information shared by individuals in this era and with the level of confidentiality demanded.

Due to ubiquitous exploitation of personal data and privacy through mobile devices and gathering of big data, new laws have been brought into action to revitalize the legitimacy of privacy in modern day. GDPR[1] is the most prominent legislation which was established in action by the European Union in 2018 filling the gaps in the earlier Data Protection Directive 95/46/EC (DPD) [16] introduced in 1995. GDPR introduces two crucial laws: **the right to withdraw consent** (Article 7(3)) and the **right to be forgotten** (Article 17). These articles restrict the data collectors from misusing the consent of the data owning individuals by sharing data and its use which leads to controversy.

Personal information collected by one organization cannot be shared with a third party for processing due to the right to be forgotten. This challenges the industry to find technical solutions to be compatible with the enforcement of the article, while the article itself does not guide through any technical aspects of its implementation. Even though the trivial solution would be to delete the data entries physically, many database and file systems are designed in ways to keep the old data while moving it to an inactive state. Such systems are facing trouble with compliance to the GDPR. Once the data is moved to different locations in other organizations tracking its origin and deleting all the copies in forgetting the data is a non transparent approach which involves effort and risk. This type of challenging scenarios puts previous ways of data protection obsolete and restrict the freedom to transfer the collected personal data. These restrictions should be addressed primarily to utilize personal data in machine learning which is often carried out by a third party other than the data collector.

2.2 Personally Identifiable Information (PII)

Since our daily life is connected closely in digital aspects in this era, various organizations can rely on our personal information for better inferences. Information about our interests through our online browser activity, our medical history, our real world connections through

interactions and so forth are shared with interested parties who might have something beneficial for us. These parties are usually advertisers, researchers and government agencies [17]. This hunger for personal information has given enough room to anticipate privacy related issues such as the ability in identifying individuals in places where it is not intended.

Defining Personally Identifiable Information (PII) would be the first step towards safe guarding the privacy of individuals. Several organizations have given definitions and have listed personally identifiable information. PII is a difficult concept in defining due to its legal and technological presence in abundance. It is defined as "Any information that distinguishes one person from another that can be used for re-identifying data" [18].

Canada's Personal Information Protection and Electronic Documents Act (PIPEDA) states that personal information includes any factual or subjective information, recorded or not, about an identifiable individual and further lists the information:

- age, name, ID numbers, income, ethnic origin, or blood type
- opinions, evaluations, comments, social status, or disciplinary actions
- employee files, credit records, loan records, medical records, existence of a dispute between a consumer and a merchant, intentions (for example, to acquire goods or services, or change jobs)

California's Online Privacy Protection Act of 2003, Data Protection Directive (DPD) of the European Parliament, and U.S. Health Insurance Portability and Accountability Act (HIPAA) are among the laws that impose regulations and keep the involved parties accountable for the protection of PII. DPD further defines personally identifiable information as "any information relating to an [...] natural person [...] who can be identified, directly or indirectly, in particular by reference [...] to one or more factors specific to person's physical, physiological, mental, economic, cultural, or social identity."

So by these definitions, commonly identifiable information of individuals when matched with databases containing highly confidential information that is not to be disclosed can bring violations to the privacy. Protection against this type of disclosure is carried out by de-identification of common attributes that can uniquely identify an individual. Methods such as k-anonymity [19] are used in this sort of de-identification work. But these methods only focus on the data itself within the specific task these data are used. An attacker can still breach the privacy by matching these data with other overlapping databases, one such example is given by [20]. In this case study, hospital discharge documents were linked with a voters' database to expose the health conditions of the individuals. The AOL search query matching

leading to a privacy debacle is another such breach known due to re-identifying anonymisation methods. A recent case study is the NetFlix price dataset exposure by matching it to the IMDB reviews hence achieving identity disclosure [11].

These re-identifications are possible due to the wider spectrum of data that is available from an individual. Computationally expensive methods are introduced in guaranteeing bound of ambiguity in identifying individuals out of a database by matching it with another closely related database to these individuals. Differential privacy is such method rigorously defined which adds noise so an identifying individuals would have a margin of ambiguity [21]. But differential privacy does not offer guarantees towards universal utilisation of the data in different tasks rather deals with the task specified to be protected by DP initially.

Due to the privacy hazards, laws like GDPR are pushing towards consent revoke, right to be forgotten and so forth in guaranteeing more transparency in privacy hence avoiding any hysteria or reluctance in contribution of data by the general public. GDPR also imposes strict guidelines on de-identified data in Article-11 regarding the personal data which further adds more responsibility to the data owner and when any third party is allowed to utilize these data. Research work suggests GDPR is encouraging research to explore methods of re-identification hence find better methods for de-identification [22].

These concerns are even stronger in the context of unstructured text data containing PII due to the challenges in applying de-identification methods. Challenging nature of this de-identification task arise from the nature of unstructured text data as well as massive effort needed for the task, and further the proneness to a high rate of error leading to an even higher rate of failure [23].

De-identification of PII in unstructured text is addressed by research mostly in the field of medical documents. These research suggest the challenging nature of true anonymisation [24] and often present results on selected sample of data which belong to few types of specific documents [25]. K-safety approach based ERASE text sanitizing system is one of such systems that suggests an entire scheme based on terms to prepare a document and then application of algorithms to anonymise, which is evaluated on 100 randomly generated documents [26]. Other methods involve replacing specific sensitive words and terms with generic ones. There are rare efforts in general research towards protecting the PII. Specific domains define the PII that has to be protected. As an example U.S. Health Insurance Portability and Accountability Act (HIPAA) protected health information (PHI) categories mention a vast number of PII that should be anonymised for de-identification of the documents:

- Names
- Geographical Subdivisions (addresses, zip codes, city and etc.)

- All elements of dates
- Images
- Telephone numbers
- Fax numbers
- E-mail addresses
- Social security numbers
- Medical record numbers
- Health plan numbers
- Account numbers
- Certificate or license numbers
- Vehicle identification or serial numbers including license plate numbers
- Device identification or serial numbers
- universal resource locators (URLs)
- Internet Protocol addresses (IP addresses)
- Biometric identifiers
- Full face photographs and comparable images
- Any other unique identifying number, characteristic, or code

Research suggests anonymization of such lists of private information strictly governed by laws without calculating the risk leads to the less utility of the data and also having to get rid of all the PII in documents as redundant. Evaluations of major challenges in this domain such as de-identification challenge in i2b2 (Informatics for Integrating Biology to the Bedside) suggests systems should be evaluated for the performance on larger and heterogeneous datasets in future [27]. Most research suggests the need of further work in guaranteeing PII safety specially in the areas of text documents. By safety we mean the protection against the breach of privacy and exposure of sensitive data of individuals. Strong access control mechanisms and non-technical methods such as informed consent for data utilization are also suggested as further protection for PII.

2.3 Automated Annotation of Textual Dataset

Though the manual annotation is costly, creating massive datasets for gaining a peak performance using deep learning models is an upfront cost in modern day machine learning research. Automated annotation of data has been a concern in machine learning to face this demand of more data and to minimize the cost and time taken for the annotation tasks. Automated annotation is even crucial in practical situations where manual annotation requires domain expertise. Privacy in sensitive data has become a barrier that cannot be worked around with manual annotation in creating massive datasets.

To gain time-wise improvements while guaranteeing the accuracy, automated annotations are done with partial manual efforts in creating annotated learning corpora. [28] presents such a semi-automated annotation method where expert annotator just directs the annotations as right or wrong. This method starts by manually annotating a part of the corpus and then proceeding with the semi-automated annotations of the whole dataset. They present a system for identifying gene mentions trained on a semi-automated dataset. This annotation mechanism can train the classifiers to achieve similar accuracy if the semi-annotated data was roughly 1.67 times that of manual annotations. A reduction of the annotation time up to 58% of the manual annotation time was achieved. This work also reduces the effort by choosing to avoid corrections to the boundaries of annotated entities.

Totally automated systems as presented by [29] gather simpler heuristics by looking into a vast number of entries in the data source. Then based on them, a set of sentences are extracted to be annotated with the class labels. They use word n-gram features and syntactic features from the selected sentences to train the Naive Bayes and SVM classifiers. Their training set contains 9985 sentences belonging to 3 classes and trained algorithm has a F1-score above 0.70 for 10-fold cross validation for different test sets. They conclude the possibility of automated extraction of a larger dataset from an unstructured text data source of scientific documents for the task of supervised learning.

A massive dataset annotation of unstructured text is presented in [30] where there is freedom to use an ample amount of alternative forms of knowledge and a vast number of data entries from Wikipedia. They use linked words in the Wikipedia text as annotated entities and assign the entity class by the categorization given to the link's destination. This approach has allowed them to do the task in a multilingual fashion.

More general approaches in the recent past have been developed in the form of frameworks for automated text dataset annotation. Snorkel is such a framework which allows many weak-supervisions (alternative sources of annotations and annotation rules) to be combined to create automatically annotated datasets [31]. Snorkel eventually produces a gener-

ative model that can produce annotated or labelled datasets from the non annotated data sources. Snorkel has implementations on unstructured text with SpaCy as its language hierarchy (Document, Sentence, Span, and Entity) maintainer where they prove the concept with the Signal Media One-Million News Articles dataset [32]. SpaCy also has syntactic parsing accuracy among 1% of the best available NLP libraries for entity parsing, ClearNLP, CoreNLP and Turbo are few to name among the comparison [33, 34].

Automated annotations are carried out for creating massive datasets as well as for specific tasks involving smaller data sources, yet popular generally acceptable datasets specific for privacy research in unstructured text data are rare.

2.4 Named Entity Recognition

Named Entity Recognition (NER) is an essential task in extracting structured information from unstructured text. For example, a specific structure of information would be a person's name, their parents names, birth place, school, employer, etc which helps in identifying them distinctly among others. More generally items identified by a proper name such as names of people, locations, institutions etc. are considered as named entities. But general expressions with numerical or temporal elements also belongs to the named entities since they are crucial in creating the structure of information carried within unstructured text [35]. For examples, a phone number, a postal code, birth date or a date of an event would fall under a to-be-recognized named entity. NER models are trained on domain specific datasets. Training on such datasets helps a model in learning the surrounding context of the entities in a sentence.

Conceptually, an NER task can be seen as identifying entities and then classifying them into the categories. Typically, the span of a name is found, segmented and then the entity type is determined. For the convenience, an NER problem is represented as a sequence of words labelled by tags where a NER algorithm tries to predict the label of each word. Several tagging schemes were invented over the time, for the convenience of representation and avoiding labelling ambiguities. Examples of popular schemes with their labels are represented in Table 2.1 for a sentence. IO scheme uses only a one tag "I" to mark the word tokens of an entity and "O" tags to mark the surrounding tokens outside the entity boundary. In the table the person entity "John J. Smith" is marked with the "I_PER" tag and location entity "Washington" has been given the tag "I_LOC" . BIO scheme differs from IO scheme by adding "B" tag to mark the beginning of the entity. Same entities mentioned previously are started with "B_PER" tag. BILOU scheme adds "L" tag to the BIO scheme to mark the end token of a multi-token entity and mark a single word entity with a "U" token. the multi-worded name entity has the "L_PER" tag as the last tag. There are research on improvements

from BILOU over BIO [36], yet BIO representation is vastly adopted in research.

	scheme and tag		
Word Tokens	IO	BIO	BILOU
Yesterday	O	O	O
afternoon	O	O	O
,	O	O	O
John	I_PER	B_PER	B_PER
J	I_PER	I_PER	I_PER
.	I_PER	I_PER	I_PER
Smith	I_PER	I_PER	L_PER
traveled	O	O	O
to	O	O	O
Washington	I_LOC	B_LOC	U_LOC
.	O	O	O

Table 2.1: Example of NER tag labelling schemes

Early days of NER tasks involved a list of heuristics for matching names and then storing a list of occurrences of possible variations of them for future events of recognition [37]. These methods were selectively focused on a narrow scope like detecting company names. But the term NER was brought to use by MUC-6 tasks [38], where information extraction tasks are generally in the form of named entities. In order to handle NER tasks that are dealing with different entity structures, supervised machine learning approaches are taken. Application of Conditional Random Field (CRF) models, Support vector machines, and extensive feature engineering based on lexical and phrase embeddings [39, 40, 36] has taken NER tasks to a higher accuracy while improving the ability to generalize on datasets with diversity. Even though these methods achieved substantial performance gains compared to more traditional ways, they still involved rather time consuming manual work such as several forms of feature engineering. Invention of word embeddings and improvements of deep learning networks have led to the utilization of recurrent neural networks (RNN) [41] for NLP tasks. RNNs were very efficient in learning NLP tasks with variable input lengths without task specific manual feature engineering. Combination of these concepts with the Long Short Term Memory (LSTM) [42] unit allowed the NLP models to encode the long-distance dependencies in context of one or more sentences. LSTM units are fed bidirectionally with input sequences making sequence label tasks like NER much more robust [43]. LSTM neural network based methods are found to be achieving higher accuracy in NER tasks just by utilizing generic word embeddings and character level feature extraction [44].

Recently, rather than utilizing only generic word vectors and training the models to learn the task out of the dataset from scratch, large scale pre-trained language models are used to get improved results in NLP tasks [45]. In simplest form, language modelling is the task of encoding sequences of words so that the model can be trained to predict the next word in a given sequence of words. This form of pre-training gives improvements in a token level task upon fine tuning for the specific task with the domain specific data, due to the pre-trained model's ability in understanding language specific rules in general. First pre-trained models were based on the LSTM as the main concept of encoding sequences [46]. While this approach is effective parameter-wise, it is inadequate to model language. Recent improvements to learn the context of a token and the formation of a sentence following a context, *BERT-base* language model has achieved state of the art results. This model is trained by two unsupervised tasks, predicting missing tokens of an input sentence and predicting the next sentence [13]. This model is trained on a very large number of generic language examples in a randomized unsupervised setting. These models have gained substantial performance increase over the past methods in solving several NLP tasks along with NER tasks [13].

Evaluation of NER tasks

Precision and recall are well suited in evaluating algorithms which deal with the task of finding a small number of named entity tokens from a large set of word tokens. Tokens are the parts entity is made up of, mostly words belonging to the entity and non entity words are also tokens. In the context of a NER task precision and recall are used as the measures for evaluation. These measures are accurate in evaluating a NER algorithm due to the nature of the task, where a large proportion of words carry negative labels compared to the very small proportion of words belonging to entities which are carrying positive labels. Positiveness and negativeness being a word is assigned a tag label or not respectively. Labels describing the desired named entities are the positive labels and others are negative labels. If the conventional classifier accuracy was taken, getting a small number of positive labels wrong or false positives (FP) can have a lesser effect on the percentage of accuracy. On the other hand marking many negative labels can give a false sense of higher accuracy due to the presence of many originally negative labels. These false negative (FN) suggestions make it unremarkable to detect a true negative.

To treat the above two scenarios separately, precision and recall [47] are used in NER performance evaluation. Precision is the measure of how much a system avoids labelling a positive token as a negative one. Precision rewards the true positive detection while penalizing on the false positive detection. False positive predictions along with the correct ones

would decrease the precision.

$$P = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{2.1}$$

Recall is the measure of how well a system can mark the positive labels that should be marked. A missing label or a false negative can decrease the recall.

$$R = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{2.2}$$

A system can forge the recall by returning every possible label for each positive token and precision by only returning a lower number of certainly correct positives [48]. Therefore, in practice precision and recall are combined to a harmonic mean by the F1 score [49].

$$F_1 = 2\frac{PR}{P + R} \tag{2.3}$$

Scores for the recognized entities by the system are evaluated with several strict and relaxed measures depending on the output scenarios of the NER systems. These relaxations and strictness are imposed based on the entity type and entity boundaries. Different shared tasks treat their system's outcome with different evaluations. Tasks such as IREX [50] and CoNLL [51] penalized the system outputs by only giving credit to the exact entity type with exact tag boundaries, which is hence known as "exact-match evaluation". Generally, these metrics are strictly penalizing the output to maintain a universal benchmark of performance. But in practice, depending on the application outputs of the systems are useful while not being accurate as demanded by the standard scoring mechanisms. In a scenario where the interest is to determine the presence of the desired entity or to quantify its presence, evaluation is carried with *type matching* or *partial matching*. It is explained in [52] how partial matching is utilized in determining sentences referring to the gene and its functional information, which is a good scoring metric depending on the particular task. Use cases in information extraction which can benefit from the systems that can determine the entity boundary regardless of the type should have corresponding metrics. Several metrics explaining their scoring elements are brought up by MUC-5 task [53] and further explained under Semeval-13 task 9 [54]:

1. **Strict evaluation**: exact boundary match and entity type as the gold labelling (labels given by human annotators).

2. **Exact boundary**: regardless of the type boundary matches exactly as the gold labelling

3. **Partial boundary**: type is disregarded and partial overlap with gold entity

4. **Type matching**: a partial matching of suggestion with gold entity while matching entity type.

We evaluate our NER model performance with all the metrics since the typical usage of strict metric would be restrictive in a task of PII recognition and comparison between the metrics also give insights into the behavior of the model. Strict metric is totally unforgiving and aims for a complete match between extracted and actual entities, whereas in practice partial matching can be useful to reduce the manual effort, while a human is still in the loop.

In order to score the precision and recall for these four metrics MUC-5 defined scoring elements are used:

- correct (COR): output and the gold annotation match
- incorrect (INC): output and the gold annotation do not match
- partial (PAR): output and the gold annotation overlap partially (Not applicable unless the metric is partial)
- missed (MIS): gold annotation is not identified by system
- spurious (SPU): system labels an entity not given in the gold annotation

Then several measures are calculated combining these elements:

- Number of correct suggestions (COR) (same value as given in the scoring elements)
- Contributing gold annotations for the final score

$$\text{POSSIBLE or POS} = \text{COR} + \text{INC} + \text{PAR} + \text{MIS} = \text{TP} + \text{FN} \tag{2.4}$$

- Number of suggestions from the system

$$\text{ACTUAL or ACT} = \text{COR} + \text{INC} + \text{PAR} + \text{SPU} = \text{TP} + \text{FP} \tag{2.5}$$

These measures are used in calculating the final precision and recall for each metric as shown by the Table 2.2

These metrics are beneficial in evaluating the NER algorithms in general. More task specific evaluation methods are utilized in practice since these metrics are not descriptive of the user satisfaction. Advanced evaluation methods based on the satisfaction of the expert users can be seen in literature, which are learning tasks themselves [55].

measure	exact/strict	partial/type	general definition
Precision	$\frac{\text{COR}}{\text{ACT}}$	$\frac{\text{COR}+0.5\text{PAR}}{\text{ACT}}$	$\frac{\text{TP}}{\text{TP+FP}}$
Recall	$\frac{\text{COR}}{\text{POS}}$	$\frac{\text{COR}+0.5\text{PAR}}{\text{POS}}$	$\frac{\text{TP}}{\text{TP+FN}}$

Table 2.2: Precision and Recall relation for evaluation schemes

2.5 Transformer Based Language Modelling

State-of-the-art NLP research has been dominated by large-scale language models. These form of models are known as transformer based NLP models. Transformer based models eliminate the difficulties in learning the dependencies between the input and output tokens in language modelling. These learning difficulties are due to input and output being related by a sequence of computations which are growing as the distance between the input and output tokens increases [56]. Transformer based models reduce this by bringing down the operations to a constant number. This transformer based method constant input positions will result in learning an averaged representation of the inputs relative to the position [57], yet this compromise has been overcome by learning multiple representations for the inputs in parallel, which is called multi-headed attention. These architectures are capable of learning language modelling at a higher capacity and naturally, demanding a tremendous amount of GPU resources [58]. These large scale models are trained on web-scale datasets in unsupervised fashion to learn a language model. Researchers with higher resource availability would train general language models in variations of BERT architectures and datasets. These models are made available so they can be fine tuned for a downstream task such as NER in our case, usually referred as a transformer. Transformers are available with various architectures or parameters due to changes in datasets. A domain-specific data source can be utilized to fine tune a general language model based on the transformers. With minor additions in the input and output layers of the network, it can be made compatible for the task and dataset. A tokenizer for the input words, and output classification layers are these additions in many NLP tasks.

2.5.1 BERT Language Model

As we mentioned earlier, to eliminate the growth of computations a transformer based model utilizes attention encoding. To make up to the loss of resolution due to position averaging a transformer based model utilizes multiple paths with the same attention mechanism which are called multiple heads of attention, and whole mechanism being named as multi-headed attention. Each attention head consist of Q, K and V sub layers (as shown in Figure 2.2)

which are contributing to the attention mechanism [59] and 12 such parallel heads are utilized in the basic pre-trained model which is called *BERT-base*. Since the same input is the input to the separate layers Q, K and V of the encoder this mechanism is called self-attention. Q, K, V are given the names query, key and value, respectively based on the intended function of these layers. They are 3 hidden layers to be learned by the encoder. Q layers outputs an input request we are attending to, K outputs the features in a query that were learnt which are to be compared with the current query, and the value V layer generates the output values. Output of the encoder network is the weighted average of the output of V layer. Weights are given by the softmax of similarity of Q and V given by the the normalized QK^T. This basic architecture is extended to larger structures in recent research [60, 3, 61].

Architecture

Embedding Layers

BERT utilizes several embeddings for input tokens. Since *BERT-base* does not have the recurrent nature of earlier models to gain positional information, *BERT-base* uses a position id embedding. This position embedding layer is a lookup layer which has a vocabulary size equal to the maximum number of tokens allowed into the network, in the *BERT-base* network this value is 512 positions. Then the sum of embeddings of dimension 768 (size of the *BERT-base* embedding output) is taken to the hidden layers as shown in Figure 2.1 by the given output "to encoder".

Batches of sentences are fed into the network after tokenizing words and padding all the sentences to an equivalent size. **WordPieces** [62] tokenization and embeddings are used in most recent models. Batches of token ids are fed into the word embedding layer. Position ids are generated by arranging a sequence of numbers covering the sentence length of the batch. Type ids are used to distinguish multiple sequences in the input tokens, which are left with a single id for NER tasks.

Encoder Layers

12 parallel multi attention heads are connected to form the encoder, each attention head with 12 attention heads. Q, K and V layers are fully connected layers with input size 768 and output size 768 as shown in the Figure 2.2; in the implementation these layers are combined to form $12 * 768$ sized single layers. Q, K and V are related by the attention mechanism in Equation 2.6 where d_k equals to the attention head size which is 64 for the *BERT-base* model. After normalisation these attended outputs are passed through a linear layer and summed with embedding outputs bypassed thorough a linear layer. Final output is normalized and

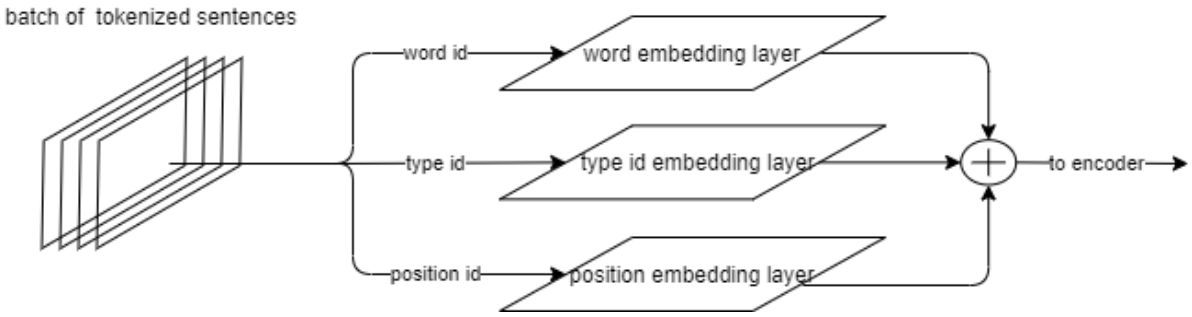

Figure 2.1: BERT embedding layers

taken through a linear layer before being concatenated with similar outputs from the other parallel multi attention heads.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.6}$$

where K^T is the transpose of K.

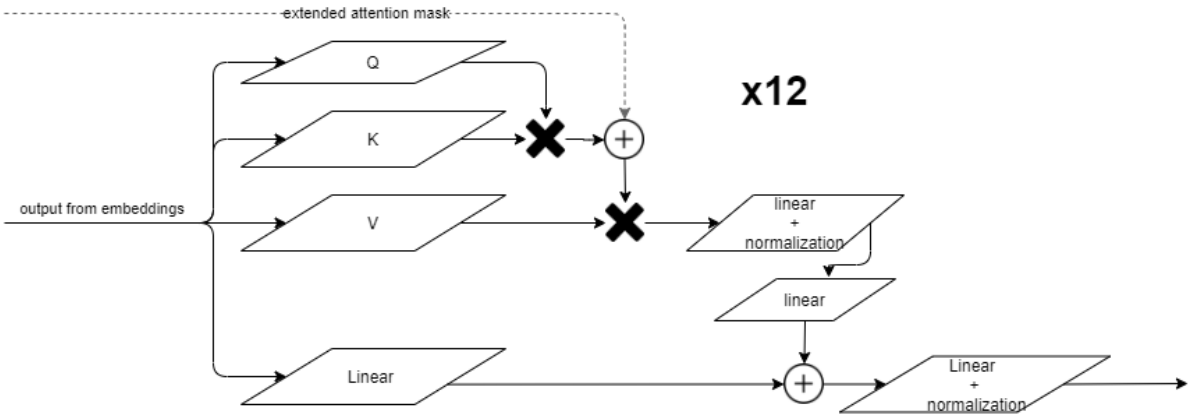

Figure 2.2: BERT encoder self attention head

Available Practical Implementations

Since pre-training of a transformer based language model is a costly operation which cannot be afforded by most researchers, several variations of BERT based language models are made available to the public by [58]. Few published models that are made available by **hugging face transformers**[1] are listed below.

[1] `https://github.com/huggingface/transformers`

- *BERT-base* created by Google and released with [13]
- GPT created by OpenAI and released with [60]
- GPT-2 created by OpenAI and released with [3]
- Transformer-XL created by Google and CMU released with [61]

These models are created by the same underlying fundamental mechanism of *BERT-base* with improvements in the training, model structure and variations in the training data.

Downstream Tasks

Pre-trained models are utilized for the downstream tasks by adding input layers needed for accepting the input data and output layers for suggesting the desired outcomes. Inputs are presented to the model with extra information that assists in the task rather than word or token ids for the input sentences. This information is integer position of the token in the input sequence called the position id and token type ids, which are giving the distinct nature of input tokens. For example, a question answering model would carry context tokens in one integer id while the question would be in another distinct id. A combination of these ids are given as the inputs to the pre-trained model and the concatenated output from the parallel multi attention heads from the model is connected to a recurrent neural network or a fully connected layer with a argmax layer to suit the final output.

As an example a NER task would take a sequence of input ids corresponding to the tokens in a sentence and output a sequence of the same length with an argmax function to suggest the entity tags ids. This can be viewed as a classification task of input tokens to named entity tags.

2.6 Federated Learning

Federated Learning (FL) [63] is the machine learning approach where the learning algorithms travel to the data geographically instead of data being stored in a central location same as the learning algorithms. Concept of addressing fundamental issues related to data such as privacy, ownership, and locality of data fall under FL.

Implementation of federated learning was first brought to life by Google researchers to enhance the models working on sensor and other sensitive data gathered by the mobile devices. Due to the confidential nature of data and activity logged by a personal mobile device gathering and storing these data in a central location is seen as a risk. To overcome

this extra responsibility, central models were updated by locally computed updates. Due to mobile devices possessing enough computational power this sort of a system was feasible in the past. This research brought to light how training algorithms have to be robust enough to handle the massive distributed nature, inadequate representation of the total distribution by each isolated source and the imbalanced magnitude of data amounts from each source. Further, asynchronous nature of updating a model with distributed workers has been brought up as a concerns for optimization infrastructure in the federated setting [64, 65]. Several methods developed for these scenarios by randomization of gradient decent are published [66, 67].

Apart from mentioned developments in order to overcome difficulties in optimization dynamics of machine learning models with FL, there is a plethora of research to guarantee security and privacy in FL. Secure Multiparty Computation (SMPC) is one such method which facilitates the protection of model weights while multiple sources can take part in Federated Learning to train the model with owned data without disclosing them. SMPC guarantees zero knowledge, where each worker with data will see the input and the relevant updates to the model based on the relevant outputs. This has been recognized as the perfect security that can be guaranteed due to its resilience against the possibilities in recovering training data from model itself due to phenomenon such as unintended memorization [68]. SMPC is utilized as the base in multiple platforms and setups of federated learning with the assumptions of certain guarantees of trust between parties and security requirements. Systems such as SecureML [69] suggests new techniques to train traditional machine learning techniques and neural networks in a SMPC setting. They further suggest efficient secure versions of nonlinear functional elements for neural networks. Sharemind [70] is another framework which tries to address the distributed nature of data and processing with SMPC by defining protocols for computation and sharing.

There are many traditional techniques combined with the distributed data to guarantee data privacy when utilized in the machine learning. Differential privacy (DP) is one of the rigorously defined techniques guaranteeing the obfuscation between the individual data points upon querying [71]. Deferential privacy is utilized in gradient based optimization of deep learning models to obscure the presence of individual entries in updated weights and relevant gradient updates [72]. In a deep learning model with differential private gradient updates, an attacker tracking the gradient updates and updated weights will have a margin of noise if attempted to distinguish the individual entries of the corresponding batch of data.

Another more secure technique that comes along with distributed training of models is the multiparty computation based on Homomorphic Encryption often referred as SPDZ [73]. Homomorphic Encryption allows all the computations to be private, which keeps the model

weights and structure private. However, these encryption based approaches results in the accuracy and privacy trade-offs due to polynomial approximations made for the nonlinear functions in machine learning models [74, 75]. Federated learning is a rather new concept for most state of the art developments in deep learning and there are many unsolved problems and generalizations of techniques to be explored such as numerical stability, security, dataset imbalance and so forth.

2.6.1 Categories of Federated Learning

Depending on the distributed nature of the data, federated learning of tasks are categorized in [76]. Horizontal and vertical federated learning are privacy guaranteeing mechanisms defined based on the separation of data elements needed in creating datasets for learning.

Horizontal Federated Learning

Horizontal federated learning applies on sources sharing the same feature space in the final datasets but resides in different spaces sample wise. Processing the data belonging to multiple retail organizations with equal patterns of customer behaviors yet relevant to non-overlapping retail goods is an example of horizontal distribution. Here, a curious party within the contributing sources might attempt to gain competitive advantages from the other contributing parties. Protection for this form of breaches by curious parties can be guaranteed by horizontal federated learning algorithms.

Vertical Federated Learning

Vertically segregated data sources share features that are common to each samples of the final dataset. This is the case when data about individuals are collected by several organizations and features from all the collectors are necessary for the prediction of a final outcome. Within this collaborative effort a sensitive information about an individual can be uncovered by matching with a public record carrying the features of the particular data point. So Federated learning solutions with secure computations are suggested for privacy preservation where data is totally invisible to a curious party.

2.6.2 PySyft: A Framework for Federated Learning

A unified framework for bringing together the above-mentioned privacy guaranteeing machine learning mechanisms was presented by *OpenMined* [2], a framework called PySyft[3]. This framework provides the platform for implementing concepts such as DP, SPDZ and remote execution by following the fundamental concept of executing tensor operations remotely [12]. As a whole PySyft has been developed with the theme **"Answer questions you cannot see"** to satisfy the ultimate goal of thriving forward for a zero knowledge proof machine learning inference.

PySyft is created to realize federated learning concepts while developing machine learning algorithms and infrastructure by utilizing PyTorch framework. PySyft is built to govern the operations in the remote locations or remote data sources by a central entity. These entities are known as workers. PySyft provides infrastructure to maintain datasets by data owning remote workers who can tag their datasets with search friendly strings. A central worker can orchestrate the training scenario where remote workers follow the commands. PySyft allows the researchers in a central location to search for remote data sources that are tagged and combine them to a federated dataset without transferring the data to the central location. To achieve this isolation of data, PySyft allows the researchers to transfer the model to the remote worker and operate remotely. Subsequently, the final model is updated by averaging weights, remote gradient updates or by sending the model to the remote datasets and train sequentially on each dataset.

To achieve this functionality, transferring model into remote locations and execution of model behavior has to be implemented. A typical neural network structure consists of parameter tensors and a set of functional operations such as activation functions, tensor representation modifications or softmax operations. PySyft handles movement of PyTorch tensors over the network to the other workers, while preserving the object structure these tensors are elements of. Every object that contributes to the neural network undergoes following steps and become ready to be sent through the network efficiently. Objects are compressed with LZ4 compression [77] by the framework. Transportation of the compressed objects and the subsequent decompression happens through a sequence of operations. At the worker who sends the objects, objects are:

1. Simplification: conversion of PyTorch objects to simple Python objects
2. Serialization: conversion of Python objects to binary streams
3. Compression: compression of the object represented by binary streams

[2] `http://www.openmined.org/`
[3] `http://github.com/OpenMined/PySyft`

These received binary streams at the receiving worker are converted back to the objects at the remote worker by:

1. Decompression: conversion of compressed binary back to decompressed binary
2. De-serialization: conversion from binary to basic Python objects
3. Detailing: conversion of some basic python objects back to PyTorch objects (Tensors)

Other internal operations of the network such as execution of activation functions and similar functional operations are propagated to the remote workers by command messages which carry function's arguments itself. Responses and results from these executions are received back to the commanding worker. PySyft is developed to abstract these remote executions under the hood as if the models are operating in a typical system.

PySyft has developed the concept of virtual workers who are residing in the same computational unit yet carries out the functionalities identically, as if they were separated by the network. This allows researchers to create a grid of workers virtually and simulate federated learning algorithms and mechanisms.

Chapter 3

Methodology

This chapter guides through the methodology developed for automated annotation task and federated learning experiments in details. Section 3.1 explains the method of programmatic annotation of the dataset and algorithms utilized. Due to our need of comparison with federated experiments and evaluation of our dataset, we experiment with centralized training under Section 3.2. For federated learning, dataset federation and the infrastructure to facilitate the transfer of model and datasets between workers had to be done, this work is explained in Section 3.3. Then the modifications in the model needed to be trained by federated learning are given in the Section 3.4. We present the experiments for the federated training in Section 3.5.

3.1 WikiPII: Dataset Creation

As one of our major contributions, we present the wikiPII dataset which contains Personally Identifiable Information (PII). This chapter follows the method involved in creating the dataset. Our primary motive for automating the dataset creation was to use it as a mean to isolate the data source hence achieving the privacy of the data source. Our annotation method looks into the data source only in ways that do not violate the exposure of individual entries.

3.1.1 Wikipedia Biographical Entries

Concerns in choosing a source for creating the dataset were: 1) data being publicly available, 2) presence of unstructured text, 3) presence of to be annotated entities in a consistent format, 4) presence of PII. We chose datasets which were available for the general public and discussed their support to be annotated automatically. We evaluated 10-K forms from

U.S. Securities and Exchange Commission (SEC) [1]. We could not proceed with 10-K forms due to the PII data to be annotated being scattered in inconsistent formatting throughout the document, which would introduce more manual effort. We decided to proceed with the Wikipedia biographical entries [2].

Biography entries follow a template [3], yet not strictly consistent with formatting as we could investigate through the scraping. Each entry consists of an unstructured text portion starting with an introduction and then arbitrarily titled with significant periods or events of the person following a chronological order starting from the early life. Significant events such as charity work, career highlights and disputes are added to separate sections. Then a summary of the person's personal information are given by the information box as shown by Figure 3.1 which tabulates the PIIs. These PIIs are our interested elements for annotating in the main text. Info box contains the birth name, birth date, birth place, spouse name, children names, parents names, education and etc. Info boxes of the entries follow a common html scheme as given by Figure 3.3 but hold subtle differences. These differences are the row name associations with the same class of entities. This leads to the need of association of various row names across all the biography entries with the entity classes for effective extraction.

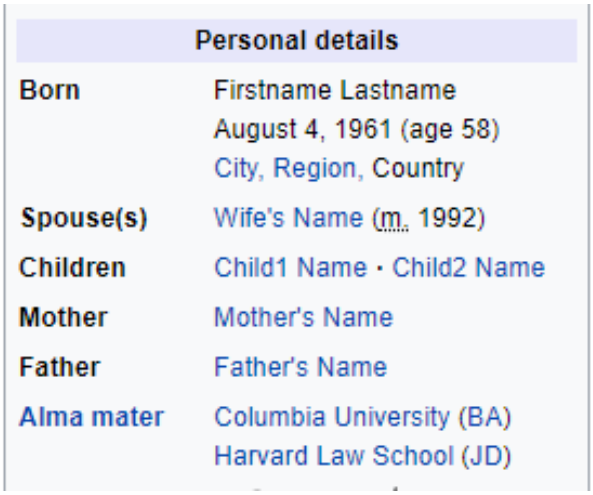

Figure 3.1: Sample Info Box view

[1] https://www.sec.gov/about/forms/form10-k.pdf
[2] https://en.wikipedia.org/wiki/Wikipedia:Biographies_of_living_persons
[3] https://en.wikipedia.org/wiki/Template:Biography

3.1.2 Automated Dataset Creation

We start with discovering URLs of the entries and proceed towards creating the final dataset as given by the Figure 3.2.

Figure 3.2: Dataset creation process

Discovering Data Entries:

We scraped the living people entries listed by the Wikipedia. Currently 949,905 entries are listed under the category of Living people[4] in Wikipedia. Links of all the entries were gathered first by crawling in the listing on the category page we mentioned above. We decided to scrape by crawling the web page itself without utilizing much sophisticated approaches such as Wikimedia Web APIs [5] due to the simplicity of scraping and the adequate ability to gather the needed data.

Meta Data Collection:

We collect metadata from the biographical entries which are crucial in gathering raw data needed for the automated annotation.

Info box is the best mean of extracting the personal data about the person described by a Wikipedia entry for tagging. While randomly looking into a number of entries, we could see a particular info box of an entry would contain only a subset of personal details we were interested in. Further similar entities were listed under different row names between entries. To extract all the entities through these variations, all the info box row names are recorded by crawling via URLs of the entries. Then these names are manually associated to the relevant classes of PIIs. These associations are given by the Table 3.1.

[4] `https://en.wikipedia.org/wiki/Category:Living_people`
[5] `https://www.mediawiki.org/wiki/API:Web_APIs_hub`

Entity	Associated row names	Tag
BIRTH PLACE	'Born' 'Born:' 'Home town'	BP
BIRTH DATE	'Born' 'Born:'	BD
PARENTS	'Parent' 'Parent(s)' 'Parents' 'Father', 'Father's name' 'Mother' 'Mother's name'	PR
SPOUSES	'Spouse' 'Spouse(s)' 'Spouses'	SP
CHILDREN	'Children'	CH
EDUCATION	'Education' 'High school' 'High school:' 'Law School' 'School' 'Schools' 'College' 'College(s)' 'Colleges' 'Alma mater' 'Almat mater'	ED

Table 3.1: Tag and Info box row name association

Figure 3.3: Info Box HTML structure

Raw Data Extraction

Info Box Scraping:

Info box scraping extracts the entities to be tagged in the text. These personal details are represented in html as shown by Figure 3.4.b. We utilise Beautiful Soup[6] for all the html extractions. We utilize Algorithm 1 to get a dictionary representation of the info box html where keys are the row names and the text elements are the data given in the right column of the table. These text elements are included in html tags depending on the cross links they carry, as an example, if a close family member is a famous person with an entry their names will be found inside a href html tag (which is an attribute that indicates the link's destination). These are removed and only the text elements are added to the dictionary. Figure 3.4 shows

[6]https://www.crummy.com/software/BeautifulSoup/bs4/doc/

a table given by Figure 3.4.a which is represented by html scheme as in Figure 3.4.b results in the dictionary given by Figure 3.4.c after the extraction. Detailed Python implementation is given in the Appendix under Section A.1.1.

Algorithm 1: Function for extracting info box html to a dictionary

Input : $Info\_box$: a BeautifulSoup html tree with infobox rows as branches
Output: $Info\_dictionary$: info box dictionary with row names as keys and text contents as entries

$Info\_dictionary = \{\}$
for $row \in Info\_box$ **do**
    $key \leftarrow row.name$ `// contents of <th> tag ex: Born, Children, Spouses.`
    $html\_tag\_list \leftarrow htmlTagListFromHtmlTree(row.contents)$ `// contents of the <td> tags.`
    $Info\_dictionary[key] \leftarrow joinTextPartsFromHtmltagList(html\_tag\_list)$ `// String content list added to the dictionary entry`

Function `htmlTagListFromHtmlTree(`*html_tree*`):`
    `/* returns a list of separated single html tags from the html tree */`
    $tag\_list = []$
    if *html_tree has branches* **then for** $branch \in html\_tree$ **do**
        $tag\_list.append(htmlTagListFromHtmlTree(branch))$
    else return $html\_tree$
    return tag_list

Function `joinTextPartsFromHtmltagList(`tag_list`):`
    $text\_parts = []$
    if *tag_list has multiple elements* **then for** $element \in tag\_list$ **do**
        $text\_parts.extend(joinTextPartsFromHtmltagList(element))$
    else return $tag\_list.append(tag\_list[0].text)$
    return $text\_list$

Private Information Extraction

Info box dictionary keys carry multiple entities, while multiple keys carry the same type of entities. These entities are organized into a separate dictionary with entity names in Table 3.1 as keys. We follow Algorithm 2 for extraction. This algorithm extracts the text parts from the info box and places them under relevant private information class based on the dictionary

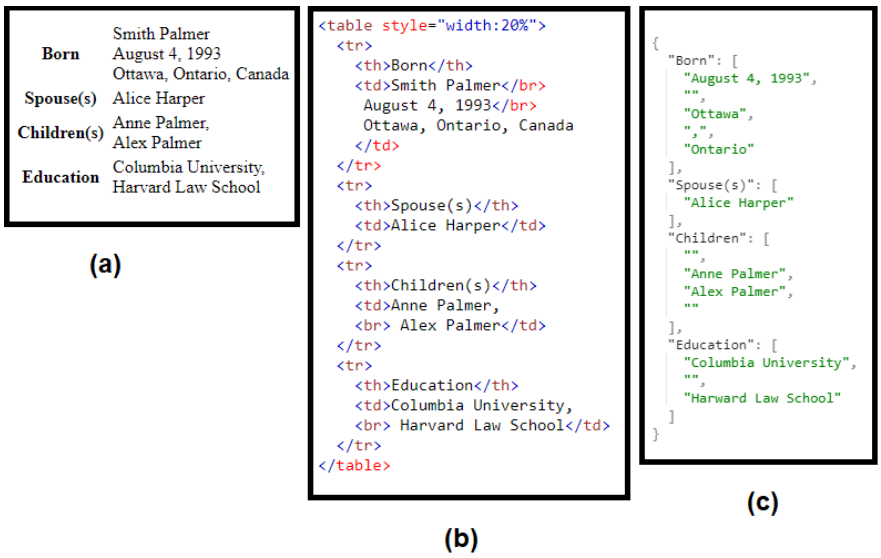

Figure 3.4: a) Info Box seen on web page, b) HTML format used to generate and c) Extracted dictionary output

key.

Algorithm 2: Function for extracting PII from info box dictionary

Input : $Info\_dictionary$: info box dictionary with row names as keys and text contents as entries

Output: $Entity\_dictionary$: PII dictionary with entities as keys

$Entity\_dictionary \leftarrow$ entity as key and the empty lists as its content

for $key \in Info\_dictionary\ keys$ **do**

  for *all* $Entities$ *associated with the* key **do**

    $Entity\_list \leftarrow$ relevant entity extracts from $Info\_dictionary[key]$

    $Entity\_dictionary['Entity'] \leftarrow Entity\_list$

return $Entity\_dictionary$

Text Scraping

Text contents under each page entry were scraped by extracting the text contents guarded by paragraph tags. Then these text parts are concatenated to form a text chunk. Extracted text from the page contains citation brackets and numerals, regular expression matching is done to remove the characters added in the text due to these citations.

Annotation

Annotation is commenced after extracting raw data, entities and text. We use SpaCy[7] on text data to maintain a structure that gives the span with indices for queries on parts of speech, named entities and sentence segments.

Entity Tagging

Entities in the entity dictionary are tagged on the sentences in the text chink. Text chunk is parsed through SpaCy. Our named entities belong to main categories; places, organizations, persons and dates. Without dissecting the data source thoroughly by looking in to the entries one by one as in a manual annotation task, which can result in violation of privacy when the data source is sensitive, we can only have general insights about the presence of entities in text. We develop entity tagging decisions based on general language manifestations. We utilise Algorithm 3 to tag all the entities but different thresholds are used depending on the entity class. These thresholds were chosen with trial and error upon running the matching on a chosen set of pages. This algorithm takes one entity from the info box with its defined class label and decides the appropriate entities extracted from the text to be matched with it, then decide if it is a valid entity tagging based on the number of matching word tokens and the fuzzy matching score. We utilize fuzzy string matching algorithm for matching the word tokens from extracted PIIs with the ones found in the text. Fuzzy matching breaks down the two word tokens to be matched to ngrams of different sizes and return the matching score which is the vector norm. This algorithm is given by Algorithm 4. A simplified implementation, an example use case, its internal representations and a sample result for fuzzy matching are given by the Listings A.2 , A.3, A.4 and A.5 respectively in the Section A.1.2 in Appendix A.

If a **person** is the general class, an entity subjected to tagging a person is treated in the same manner for all the classes in the dataset. Attributing to SpaCy's higher parsing accuracy we assume noun chunks from SpaCy to include desired person names to be tagged without missing. SpaCy's noun chunks returned higher number of person names than when queried only for the person name entities. So we first try to find the matches in person entities returned by SpaCy then try the same matching function on noun chunks from SpaCy. Then we take the union of both groups of matches. By random checks on the entries we were able to observe names on the info box were presented with certain variations to that of exact person name mentioned in the text. This left us with implementation decision to match the

[7] `http://spacy.io/`

Algorithm 3: entity tagging

Input : $Entity\_list$: a list of extracted entities from info box of same type
Output: $Tags$: a set of tags for the entities in the list as they are found in the test
$Tags \leftarrow \{\}$
$Entity\_chunks \leftarrow$ {all the named entities in the text with a similar class to $Entity\_list$ ex: date, organization, person}
$Noun\_chunks \leftarrow$ {noun chunks found in the text}
for $chunk \in Entity\_chunks \cup Noun\_chunks$ **do**
    $Fuzzy\_set \leftarrow$ {each token in the $chunk$ } `// added to fuzzy string search algorithm`
    for $Entity \in Entity\_list$ **do**
        $Token\_scores \leftarrow$ [fuzzy matching scores for all the tokens in the $Entity$]
        if *number of matching tokens and their scores exceed the threshold* **then**
            $Tags \leftarrow Entity$;

Algorithm 4: Fuzzy string matching

Input : $String\_list$: String list
$Candidate$: Matched string
Output: $Results$: A list of tuples with $String$, matching score pairs
$String\_Dictionary \leftarrow \{\}$ Dictionary that keeps ngram dictionaries for $String\_list$
for $String \in String\_list$ **do**
    $Dictionary\_list \leftarrow []$
    for *desired ngram size range of ngrams of* $String$ **do**
        $Ngram\_dict \leftarrow$ ngrams of $String$ added as a reverse lookup dictionary, where values are the presence
        $Dictionary\_list.append(Ngram\_dict)$
    $String\_Dictionary[String] \leftarrow Dictionary\_list$
$Candidate\_Dictionary\_list \leftarrow []$
for *desired ngram size range* **do**
    $Ngram\_dict \leftarrow$ {count of ngrams of $Candidate$ added as a reverse lookup dictionary}
    $Candidate\_Dictionary\_list.append(Ngram\_dict)$
for $String \in String\_list$ **do**
    for $ngram\_dict \in String\_Dictionary[String]$ **do**
        $norm \leftarrow$ compare the $ngram\_dict$ with the relavant entry in $Candidate\_Dictionary\_list$ and get the norm
        $Results.append((String,$ maximum value of $norm))$
return $Results$

First name from info box entity with the extracted entity from text. This resulted in incorrect matches for certain entries but generally treated the majority of entries fairly. Also often mentioned first names would not be the exact first name and had a shortened form in the person entities extracted from the info box. An example of this difference in mentioning names in the info box and the text can be seen in Figure A.13. We proceeded with the fuzzy string matching along with an assertion on matching character length.

An organization when represented by the entity of interest in tagging, is treated only as an organization entity. We rely only on the organization names returned by SpaCy as the candidates to be matched. Organization entities of our interest are the education related entities extracted from the info box. SpaCy identifies these entities as organisations with a high probability. These organization entities are universities, colleges and schools which are listed as the educational institutions. If fuzzy matching score between the word tokens in the entity extracted from the info box and each word token in the organization entity from text is over a threshold we consider such a match as an eligible annotation.

Birth dates are tagged by matching extracted date entries from the info box with the date entities from the text chunk returned by SpaCy.

Our system can extract the **birth place** from the info box but places are authored in several variations such as town, province, country or just the relevant place or only the country. We omit this entity in the final tagging due to a vast number of misplaced tags for the birth place if we fail to find the exact match.

Annotations are stored as the word token span. Span is represented by indices of the word tokens.

3.1.3 Dataset Creation

Annotations are initially stored as the csv files where entity tags are marked with the IOB scheme. We utilize the tag vocabulary in Table 2.1 suffixed by B and I tags. We separate training, test and validation sets based on the entity presence score of the entries. This entity presence score is decided by the presence of the 5 classes of named entities we were interested in. We ensure that entries with all the interested entities or scored 5 were added to the test set which are; date of birth (DB), names of parents (PR), names of spouses (SP), names of children (CH), names of education institutes attended (ED). Then we added the entries scoring 4 as the validation set and rest of the entries scoring at least a score of 3 and below as the training set as summarized in Table 3.2. We utilize only the sentences containing our desired tags in the training.

We store our final dataset in text files consisting of word and tag token pairs. Details for

Score (no. of entity classes)	no. of entries	contributing dataset
5	307	test
4	2745	validation
3 2 1	7853 6284 4344	training

Table 3.2: Entity presence score for Wikipedia biography entries

the final dataset are given under the Section 4.1.

3.1.4 Manual Annotation

We only utilized raw data from the info box for annotations. As we presented, for annotation we were relying on this raw data extraction and string matching algorithms running on the noun and entity extractions from the SpaCy. Due to this nature we are left with two major concerns:

1. Differences in entity instances in the full text and raw data of info box
2. Inconsistencies between data source entries

These occurrences detrimentally affect the quality of the annotations and the size of the dataset respectively. To investigate this impact we compared a subset of the automatically annotated dataset with manual annotations.

To create this manually annotated subset, we chose a substantial number of entries from the test set and employed an annotator to specify the entities of interest. Then we calculate the evaluation metrics that were explained under Section 2.4. Manual annotation is done by re-annotating the entities already found by the automated annotator. Manual annotation utility was designed this way to utilize less annotation effort while evaluating found entities. The annotator was shown the info box elements with the corresponding info box row names to know the entity associated. An example of the annotator user interface is shown in Figures A.1, A.2 and A.3 under Appendix A.1.3. The annotator has the freedom to correctly re-annotate the highlighted entities with correct boundaries and class name or to avoid re-annotating if it is a false recognition where the particular annotation is discarded in the final result.

These annotations are used as the manual test set. We present the results for the comparison of manual annotations with their automatically annotated counterparts in Section 4.1.2.

3.2 Centralized Training of NER on the Dataset

We extend the transformer based pre-trained model for the NER task to be trained with our dataset. We utilize the vanilla implementation of the **BERT-base** model for our task.
We fine-tune with our WikiPII dataset and test the model with automated and manual test sets.

Training Configuration and Hyperparameters

The configuration of the downstream model based on *BERT-base* developed for the CoNLL-2003 in achieving state of the art accuracy was adopted. We choose a batch size of 128 sentences and a maximum length of 50 tokens as a fair limit for sentence length. We present results for an epoch of training. Results are presented under Section 4.2.1 for the centralized training.

Experiments

We record the performance of the model in this typical centralized setting for CoNLL-2003 dataset as a benchmark for the subsequent experiments with federated learning. We record the performance of our WikiPII dataset trained model on two test sets, automatically created test set and the manually created test set. These results are taken to compare and explain the test scores on the automated test set, and eventually, to get the measure of success of our automated training dataset creation approach when combined with a transformer based model to train a NER model.

3.3 Dataset Federation

We create a scenario where a dataset is distributed in isolated islands. Infrastructure was developed to carry out training while abstracting this distributed nature. Infrastructure is the virtual distributed environment consisting of the mechanisms to transfer the model and data between workers and to maintain the virtual workers. For this, we developed the federated data iterator. A federated data iterator is the utility which allows us to iterate through batches of data distributed physically as if the data was loaded into a one central entity.

In the centralized training our input words are tokenized by the pretrained tokenizer and converted to token ids before feeding to the network's embedding layers. To alleviate the complexity in remotely operating the word tokenization and token id look up, in the federated setting, we convert the distributed data sets to integer token ids. Our final system is given by

the Figure 3.5. Here, the central worker governs the system's functionality. Remote workers are waiting to receive and execute the model on demand or to send data to the central location on demand upon interactions.

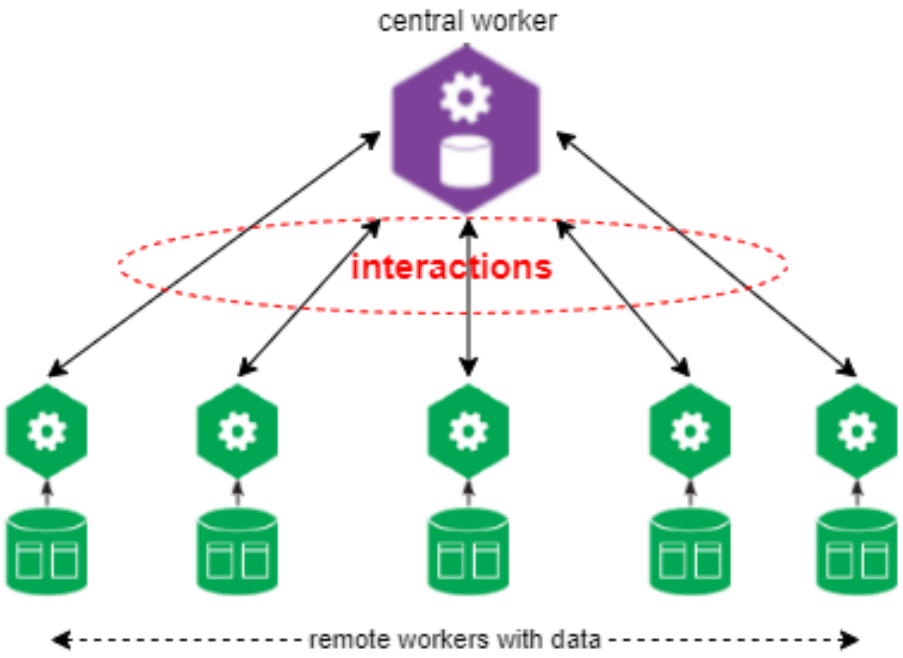

Figure 3.5: Federated system with isolated data sources

3.4 Extending BERT-base for Remote Execution

As we described, PySyft has the capability to centrally govern a sequence of operations on remote tensors which is the primary requirement for remotely executing a model which benefits in isolating a data source. Yet, there are operations in practical model structures that are not realizable by this framework currently. These operations are demanding the remote tensors to be called to the central worker hence using their dimensions to dynamically decide the internal dimensions of the model, where total isolation of the dataset gets violated. Such operations are avoided in our model by modifying the structure to become non-dynamic by imposing agreements between the model and the isolated data source.

Changes in the Model Structure

A BERT model operates by loading weights from the larger pre-trained language model which is sandwiched by its input layers and output layers developed to suit the downstream task. But, often times these downstream models would not require the inputs that are used

in the pre-training. Further, inputs will be calculated based on the dimensions of the primary input. In our model, these are the token ids from the wordpiece tokenization. In our NER model inputs are:

- token type ids: when more than one sequence is fed, tokens of each sequence will be given a distinct id. For example, for a question answering task, it will be as below where question and answer are given two distinct values 0 and 1 as given below, [CLS] tag indicates the beginning of a new sequence and [SEP] tags indicate the internal separations of the sequence

tokens: [CLS] is this jack son ville ? [SEP] no it is not . [SEP]
type_ids: 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1

  These ids are given as input data in relevant tasks, or generated as zero tensors in other tasks

- position ids: each token will contribute a number representing its position in the sentence

- input ids: dictionary id given by the tokenizer

- extended attention mask: token masking information to the model

Except for input ids, all the other inputs are dynamically generated tensors at the training time for the downstream NER task to function using the pretrained *BERT-base* model. These inputs are generated by the off-the-shelf model to be compatible with the dimensions of the input sentence batches. PySyft framework cannot query these dimensions of the input data tensors when demanded by the model, while operating in a remote worker due to the input data being situated remotely.

This restriction is due to architecture of the framework being built to abstract the remote tensor objects by wrapping them around an empty tensor located centrally. Wrapping the tensor is the process of maintaining an empty local tensor object while necessary methods of it are taken to the remote tensor through the network. This method abstracts the location separation and allows the central worker to operate on tensor objects just as they were situated centrally.

One drawback of this wrapping based abstraction is the functions such as size querying are operated on the local empty tensor rather than the native real tensor situated in the remote worker. An example of an empty tensor shadowing a remote tensor is shown in Figure 3.6 which is resulted from moving a tensor from Alice to Bob.

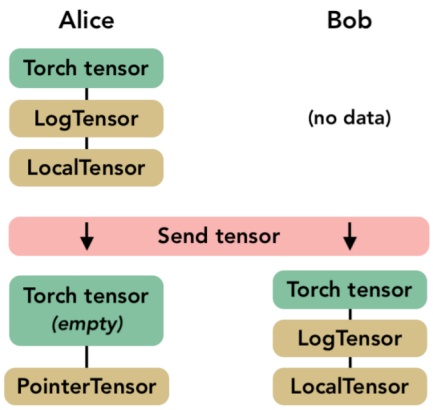

Figure 3.6: Tensor movement between two workers as given in [12]

A possible workaround to operate our model with these internal dynamics and the framework APIs is by calling the data into the central location and querying the dimensions which violates the isolation of the data source. We eliminate this data calling requirement by changing the dynamic nature of the model to be static, where central worker knows the input batch dimensions before sending the model to the remote location. Remote data holder agrees to maintain input data batches matching these static dimensions of the model. Yet, these input has to be generated and sent to the remote worker at run time which leads to additional interactions with the remote worker. To support this, we modified the model to carry these inputs as static non-trainable parameters embedded in the form of tensor buffers. Using PySyft we can move a model between the remote workers and the central worker using the API calls. Initially, these APIs were developed to handle the movement of trainable parameters of the models among workers involved in federated learning. But for our new mechanism APIs had to be extended to handle the movement of non-trainable parameter tensor buffers between workers. We contributed to the PySyft framework's code base by extending its APIs to handle the movement of non-trainable parameters, hence achieving the total remote functionality of the model.

Figure 3.7.b depicts the operation of the remote model on an already tokenized data from a remote source in contrast to the typical operation as in Figure 3.7.a. The calculated additional embedding layer inputs are integrated together with the model parameters as explained to facilitate remote execution.

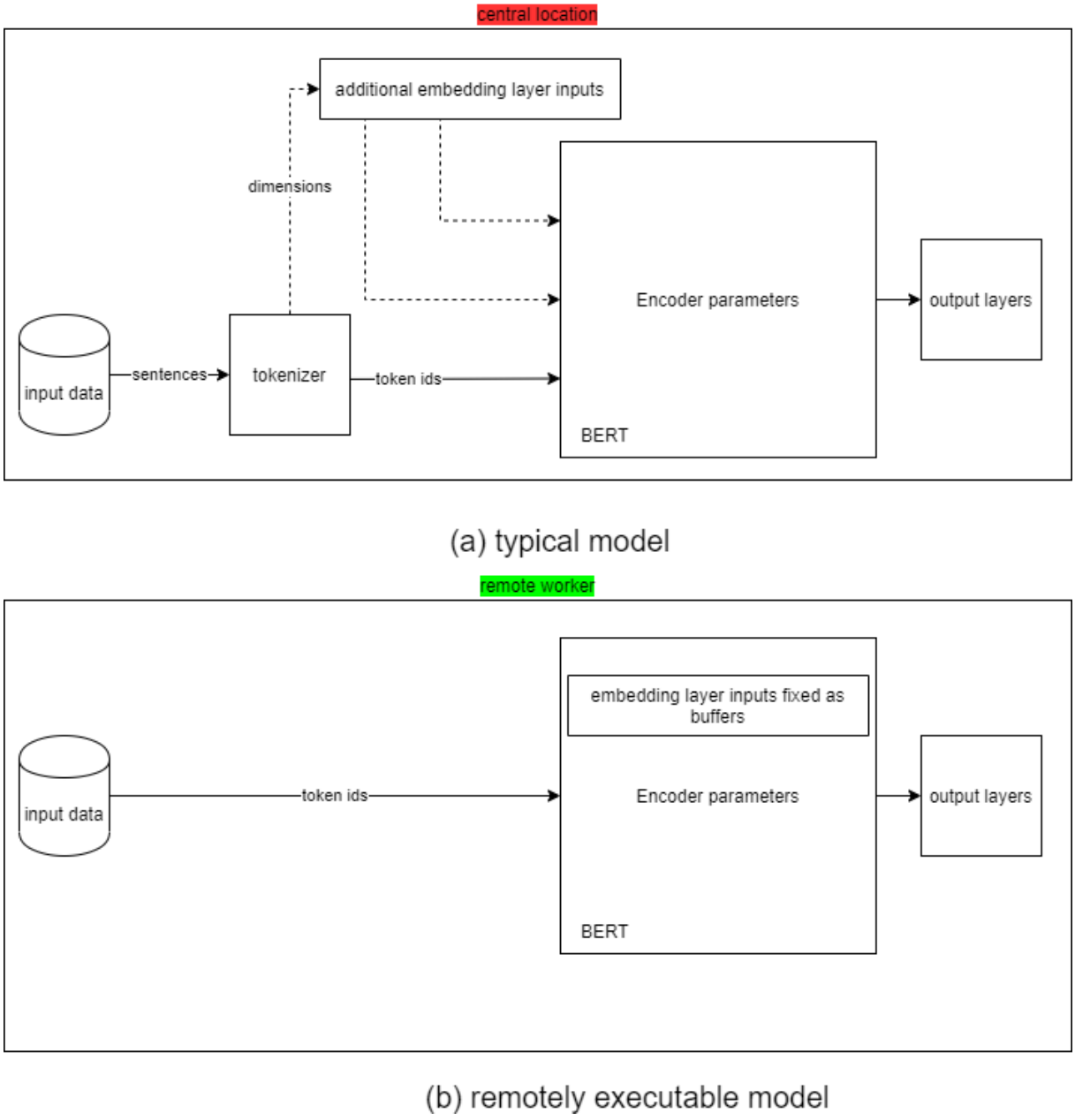

Figure 3.7: Modification of the model: broken arrows show the dynamic tensor inputs

3.5 Federated Training

We have picked two settings of federated learning that can address the privacy preservation by data source isolation while depending on the resource required and the timeliness. Here, resources required mean the computational resources required for running a virtual federated learning system. For example, if we were to run a weight averaged training scenario where simultaneous model training happens and the final model weights are formed by the average value of weights of these models, the demand of resources will be multiple of the presented experiments. Timeliness is attributed to the techniques finalized by the community in the framework. For example, if we were to use fixed precision representations and encrypt the weights of the model, we had to put efforts that are beyond most expertise in the open source community who are developing PySyft and would not yield much importance.

This chapter details the experiments done in both federated training settings for the datasets WikiPII and CoNLL-2003. To present generally comparable insights for the changes in results of training due to federated learning, we trained the same models on CoNLL-2003 dataset. We restrict our experiments to one epoch of training. By this we make our final results for all the training scenarios, central and two federated settings comparable agnostic of the iterations. We also assume the availability of the remote workers at all the times a central worker requests their computational resources for training, which might not be the real world scenario and needs planning and robustness which are beyond the scope of this thesis. Our training results are taken for two scenarios:

1. Trusted operator: receiving data from remote data holders (federated/central)
2. Mistrusted operator: sending model to a remote data holders (federated/remote)

Both scenarios have their use in privacy preserving by isolation depending on the trust given by the central operator. But they come with inherent drawbacks compared to the centralized training. To investigate the impact on model performance we recorded the convergence of the training loss and test set accuracy on the auto-annotated dataset and CoNLL-2003 dataset. We present the results in Section 4.2.2. We present results for the scenario of simulation with 2 virtual workers, due to the observed impact which is the drop in accuracy not being affected by the number of workers.

3.5.1 Training on the Federated Dataset by a Trusted Operator

In a scenario where the operator is trusted to receive data from remote sources, the model can be operated in the central location. Data batches from distributed sources are called using

the federated training iterator. Received data batches contribute in forward pass and back propagation to train the model and afterwards the operator discards the data batch as agreed. Model can work as in the typical scenario of training without internal changes as mentioned in Section 3.4 due to operators unrestricted view into the the data batches. Interactions in the system are data transfers from the remote workers towards the central worker.

3.5.2 Training on the Federated Dataset by a Mistrusted Operator

In this setting, the operator is not fully trusted as in the case of calling data to the central operator. Federated training iterator holds the locations of the data holders or remote workers. The model owned by the central operator is sent to the remote worker and allowed to be remotely executed. Only the commands from the central operator are allowed to reach the remote worker guaranteeing the central operator is not breaching into the data. Interactions for this training according to the Figure 3.5 are sending model, sending commands to execute the model and receiving the trained model parameters back. Model weights are received back by the central operator after training for all the batches of data belonging to the remote worker. Then the model is sent to the other workers. An epoch is completed when the model cycles all the workers in this manner. We recorded the training loss and the subsequent test accuracy for one epoch of training to investigate the impact on model convergence and the generalization due to the transportation of the model to the remote worker. Recorded training loss values are used to plot the error plots and test accuracy is tabulated in the Section 4.2.2.

3.5.3 Impact of Dataset Size

One of our interests was to investigate how well this method would benefit by its distributed nature where more data points from a number of federated datasets are utilized in training. In reality different distributed sources contributing in training can carry imbalanced amounts of data and features. Being robust to these scenarios is application dependant and beyond our scope. We create a scenario where 10 workers divide the training dataset among them and training on one worker is 10% of data, two workers being 20% and so forth. Our goal of this experiment was to investigate the improvement of the model performance on the test set vs. the increase in dataset size by the contribution of data owners. We present the results in Section 4.2.3.

Chapter 4

Results and Discussion

This chapter documents and discuses the results of:

- Details of the dataset
- Automated annotation accuracy compared with the manual annotations for the test set of the dataset
- Results of the state of the art model's performance tested with the automatic and manually annotated test sets
- Results for federated training scenarios for WikiPII dataset and CoNLL-2003 dataset

4.1 Results of Dataset Creation

4.1.1 WikiPII Dataset Summary

Separate splits of the created dataset and numbers of entities contained are presented in Table 4.1. These dataset splits were decided by the score given to the entries based on the number of classes of entities they contained. Class names are BD (date of birth), PR (names of parents), SP (names of spouse(s)), CH (names of children) and ED (names of education institutes attended) which are also given in the Table 3.1.

Discussion

Our source contains over 900000 entries, of which our annotation mechanism could only use little over 23000 entries due to formatting changes in the Wikipedia pages where infobox is not available. The dataset contains a larger number of PII instances belonging to over 23000

Dataset	Entries	Sentence count	BD	PR	SP	CH	ED
training	20039	77703	16883	6326	25163	10824	24365
validation	2744	12267	2512	1509	3844	1846	3831
test	307	2051	303	331	609	604	534
test (manual)	91	320	76	50	80	62	92

Table 4.1: Dataset Summary

individuals worldwide belonging to 5 classes. Similar automated annotation tasks such as [30] are utilizing a broader set of Wikipedia pages and contain general CoNLL style [51] (LOC(Location), ORG(Organization), PER(person), MISC(Miscellaneous)) classes of entities and does not explicitly focus on granular personal information as ours. These methods rely on Wikimedia resources compared to our infobox-only approach.

4.1.2 Comparison of the Automated Annotations

Results for the comparison between automated annotations and the corresponding manual annotations are presented in this section. We considered manual annotations as the gold standard to calculate precision, recall and F1-score of automatically annotated data set in measuring the quality of automatic annotations. Results are given in the Table 4.2. These manual annotations are the manual test set used in the final model evaluation. Section A.1.4 presents few ambiguous annotations by the automated system when compared with the manual annotations.

	type	partial	strict	exact
correct	243.00	169.00	166.00	169.00
incorrect	6.00	0.00	83.00	80.00
partial	0.00	80.00	0.00	0.00
missed	120.00	120.00	120.00	120.00
spurious	279.00	279.00	279.00	279.00
possible	369.00	369.00	369.00	369.00
actual	528.00	528.00	528.00	528.00
precision	0.46	0.39	0.31	0.32
recall	0.65	0.57	0.45	0.46
F1	0.54	0.47	0.37	0.38

Table 4.2: Automated annotation accuracy compared to manual annotations

Discussion

We discuss the comparison between our automated annotation and manual annotation given in Table 4.2.

We observe that a larger number of **spurious** annotations (279 out of 528) are done by our automatic annotator. These are the annotations which are not approved by the expert annotator while performing the manual annotation. These annotations are most probably the entities which are mentioned in the info box and appearing in a sentence without a matching context to the given entity class. For example, an education institute can appear multiple times in the text where it is mentioned to explain a non personal event or mentioned not to explicitly describe the person's education history. This **spurious** annotation behavior of the automated annotator adds noise to the training set mostly as incorrectly annotated classes. Section A.1.4 gives a set of examples of ambiguous annotation scenarios.

Missed count of the annotations holds the next higher value in the comparison. These are the entities that are present in the text closer (nearby sentences or in the same sentence) to the already identified entities but missed by the automated annotation probably due to not exceeding the matching thresholds or not being extracted from the info box. These entities are easily read and annotated by the human annotator.

The **strict** metric identifies 83 entity annotations as incorrect annotations. This is the count of the automated annotations that do not match on both class and token boundary. But when we look at the **type** metric column, we observe only six incorrect annotations. This is the count of entities not matching on the type given by automated annotations, which tells us automatic annotation has failed to annotate 77 of the entities with correct token boundaries. This observation shows that our training data includes high noise in boundaries.

These factors affect the final F1-score of the annotations under all the metrics. This is also an indication of the nature of introduced noise to the training data. We can observe the model's performance has improved on the same manual test set we compared our automated annotations to as given by the Table 4.3. When considered with the final training results our annotation method gives ability to train the model to get effective results in recognizing the entities which is to 0.80 in F1-score.

4.2 Results of Model Training

4.2.1 Centralized Training Performance

We present the results from the model fine-tuned by centralized training on the created training set. To compare the training performance, we tested the model on our manually annotated

test set. Results averaged from 10 trials are given by the Tables 4.4 and 4.3.

	type	partial	strict	exact
correct	292	206	204	206
incorrect	3	0	92	89
partial	0	89	0	0
missed	72	72	72	72
spurious	73	73	73	73
possible	367	367	367	367
actual	369	369	369	369
precision	0.79	0.68	0.55	0.56
recall	0.80	0.68	0.56	0.56
F1	0.80	0.68	0.55	0.56

Table 4.3: Test accuracy of the model on manual test set

	type	partial	strict	exact
correct	1611	1632	1489	1632
incorrect	177	0	300	156
partial	0	156	0	0
missed	583	583	583	583
spurious	526	526	526	526
possible	2370	2370	2370	2370
actual	2314	2314	2314	2314
precision	0.70	0.74	0.64	0.72
recall	0.68	0.72	0.62	0.69
F1	0.69	0.73	0.64	0.70

Table 4.4: Test accuracy of the model on automated Test set

Discussion

Results from the centralized training as given by the Table 4.3 show final model performing better on the manual test set in identifying correct entity types with 0.80 F1-score in **type** metric, but **exact** and **strict** scores being low, 0.56 and 0.55 respectively. This suggests our dataset helps the model at learning contextual information about the entities while being resistant to noise in the form of incorrectly tagged types as discussed in Section 4.1.2. It can be observed that a low number of **missed** and **spurious** suggestions from the model on the test set (72 and 73, respectively) is repeated across all the metrics. This repeated pattern can be interpreted as the model's ability to learn from noisy training data and improve not to repeat the same mistakes in incorrectly suggesting the classes as in the training set. A

portion of the **incorrect** labels under **exact** has been recognized as the correct suggestions under the **type** metric, 89 incorrect suggestions under **exact** reduces to 3 under the **type** metric. This is the number of entities identified with errors in token boundaries yet having the correct class. We can observe improvements in the F1-score for **strict** and **exact** metrics when compared with the annotations in Table 4.2 but not as much as the improvement in **type** metric (0.54 to 0.80) this is due to a substantial number of **incorrect** suggestions (92 and 89). This observation demonstrates the model's less ability in learning to suggest correct boundaries from the noisy training data.

We observe a lower **type** score in the automated test set than the manual test set which is 0.69 compared to 0.80 in Table 4.4. Here, if we consider manual test set to have zero noise then this **type** score difference is an indicator of final model learns to identify types correctly despite the type noise in the training dataset. But **exact** and **strict** scores don't suggest any form of boundary noise rejection in evaluation, due to the model giving a high boundary scores to the noisy test set while giving a low boundary scores to the manual test set. This can be seen by the **strict** metric F1-score being 0.64 and 0.55 respectively. Our dataset creation approach is beneficial at creating a large training set that can train a NER model to identify entities in text. But learning to extract exact entities with this dataset falls short.

Automated test set was created to be a dense set of annotations out of all the entries containing 5 classes of the entities. Given the model's performance on the manually annotated test set in recognising types and the **correct** count under **strict** metric, it can be observed as a less probable event to fit into a random type and a random token boundary at the same time when compared with the performance on the manual test set. This suggests a large proportion of **correct** suggestions under **strict** metric should belong to absolutely correct class (type). By absolutely correct classes, we mean that class type is correct and would be identified the same if annotated manually.

Regarding the utility of our dataset under these metrics, a de-identification task that masks the private entities would perform poor with this given performance when seen with our **exact** metrics. A de-identification task needs a 100% accuracy and has to be carried out manually to meet this criteria. A system that suggests annotations to reduce an expert's effort in masking can benefit with our performance level in identifying a majority of entities present. Our final performance of 0.8 F1-score is obtained with a training dataset created without any significant human interactions with the text as explained in Section 3.1. This dataset creation combined with a transformer based NER model can be seen as a successful way of preserving a substantial utility of the data source in learning a NER task while confidentiality of the data is secured. This approach eventually leads to the privacy of PII.

4.2.2 Federated Training Performance

Results of experiments explained under Section 3.5 are summarized in the Table 4.5, Figure 4.1, Figure 4.2 and Figure 4.3.

Filled line error plots are representative of the deviations observed in the average of 10 trials carried out. F1-scores for the strict metric for model under different scenarios of federated training are given by the Table 4.5. Figure 4.1 and Figure 4.2 show the convergence of the loss when model trained on two federated scenarios compared to the typical centralized training. We only draw the plot between federated/remote model training and the centralized model for WikiPII dataset. An observable deviation is not present between the federated/remote model training and central training which is similar to Figure 4.2.

dataset	training setting	no. of workers	F1 score
CoNLL2003	central	N/A	0.90 $\pm$0.005
	federated/remote	2	0.85 $\pm$0.003
	federated/central	2	0.90 $\pm$0.008
WikiPII	central	N/A	0.70 $\pm$0.006
	federated/remote	2	0.56 $\pm$0.02
	federated/central	2	0.70 $\pm$0.01

Table 4.5: F1-score for central vs federated model after 1 epoch

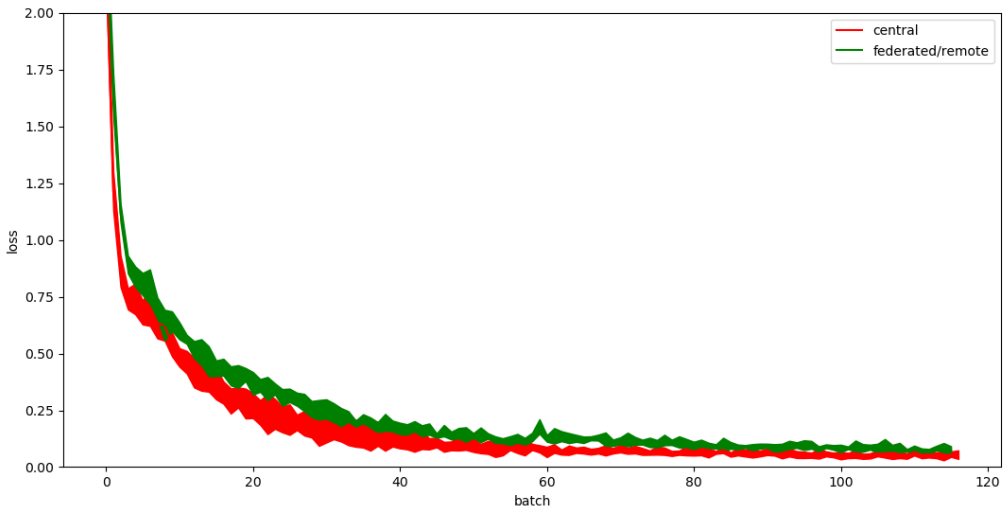

Figure 4.1: Training loss: central vs federated/remote model with 2 workers on CoNLL-2003

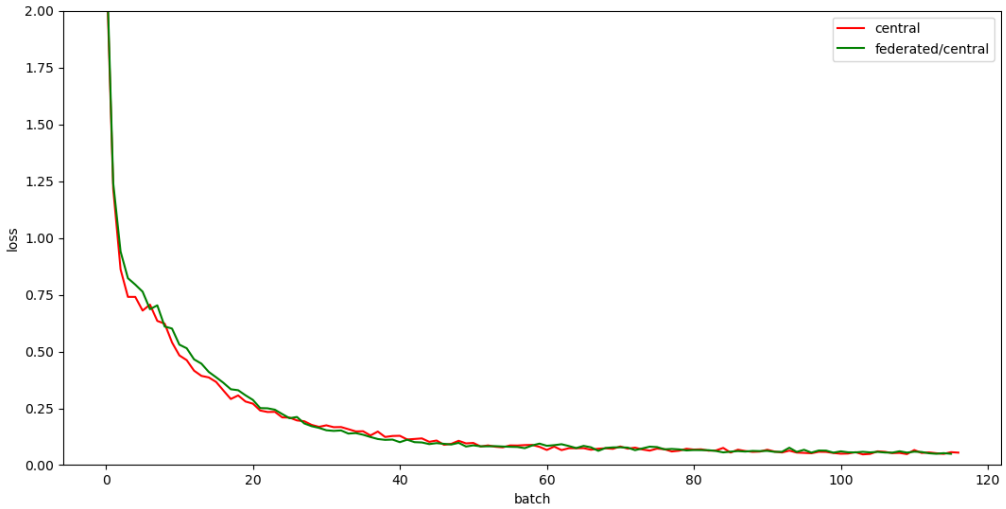

Figure 4.2: Training loss: central vs federated/central model with 2 workers on CoNLL-2003

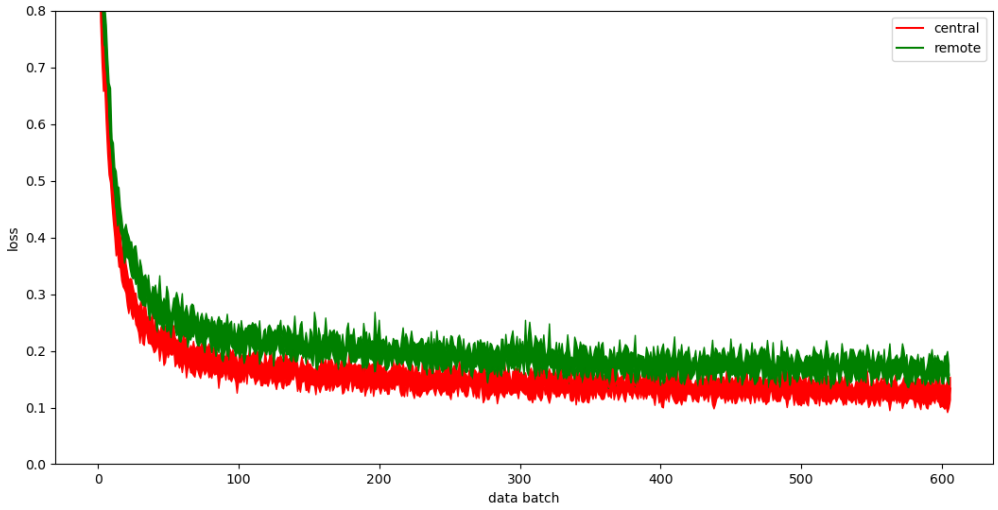

Figure 4.3: Training loss: central vs federated/remote model on WikiPII dataset

Discussion

From our two scenarios of federated training where the central operator is trusted and not-trusted, a major functional difference was transferring the parameters of the model between workers and the central operator. We can observe an identical convergence behavior of training loss (Figure 4.1) and very close final model performance (Table 4.5) between the federated learning with centrally operated model and the typical centralized training. Federated training with the mistrusted central operator deviates from both these scenarios by showing higher values in loss convergence (Figure 4.2) and a reduced final performance score.

We can account model transfer as the only functional difference for this deviation. In model transfer, all the parameter tensors of the model go through the simplification, serialization and compression steps followed by decompression, de-serialization and decompression steps. We suspect this mechanism affects the precision of the weights. A rigorous analysis should be carried out to verify this effect while utilizing PySyft for federated learning. This sort of object transfers are inevitable when having to operate in a distributed system. For federated learning to be robust while operating under these dynamics, the developed models should account for such occurrences.

We discover the loss of accuracy in remote scenario compared to the central scenario which is possible due to the compression of the weights while transferring them. This is an unknown occurrence in the PySyft framework. In our system, this deviation was observed due to the two distinct simulations we carried out. This form of a simulation can be utilized for investigating the effects of transfers on the final model performance.

4.2.3 Results on Investigating the Performance vs the Dataset Size

We present the results of our experiment explained in Section 3.5.3 where the dataset is distributed to 10 federated workers. Collected model weights from separate training runs are loaded to evaluate the test set performance. Results from 10 trials of training with each training set percentages were averaged; the error plot in the Figure 4.4 demonstrates the final results.

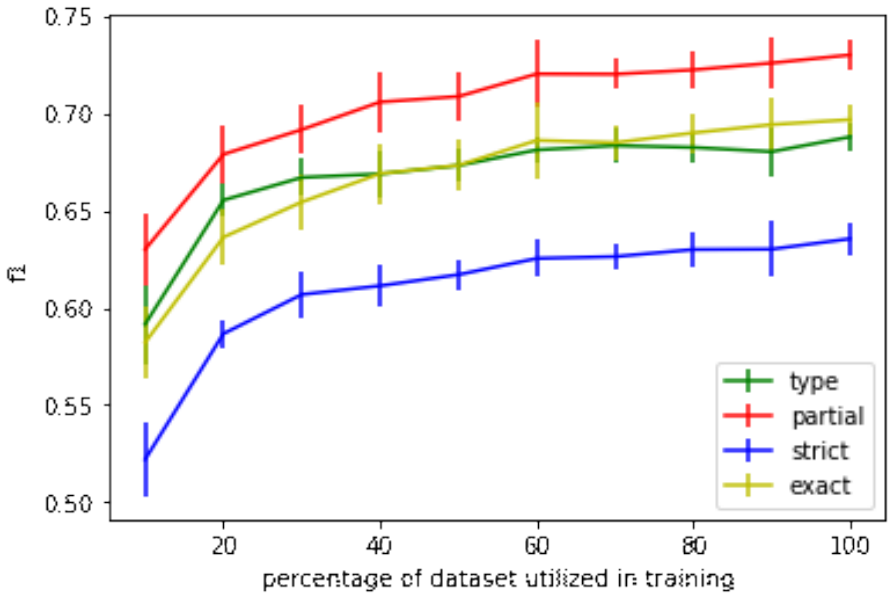

Figure 4.4: Test accuracy vs. the training dataset size

Discussion

With the increase of the size of the dataset, it is observed by the Figure 4.4, the model performance on the test dataset has increased under a constant number of training iterations (one epoch). By utilizing large federated datasets distributed among many isolated sources a model can learn faster with a minimum number of interactions with one data owner.

Our experiments show increasing volume of a programmatically annotated dataset can results in reduction of cost for obtaining manually tagged sensitive data and increasing data volume also make up for the noise in annotation leading to the improvement of accuracy. This nature helps in obtaining highly sensitive data for training models with a less risk and a low cost of annotation while yielding a satisfactory performance in the final model.

At a glance, this experiment shows the ability of automated annotation and the increase of dataset volume by federated learning leading towards the proportional increment in the performance of the NER task.

Chapter 5

Final Remarks

5.1 Conclusion

Our overall setup of automated annotation and federated learning shows the possibility of preserving data privacy by isolation of the data source while utilizing it for learning a NER task. Wherever we have access to structured data (such as the info bar) alongside their corresponding unstructured text, programmatic annotation can be used as a reverse engineering method for generating noisy annotations for training purposes. Our experiments show feasibility of this approach. With larger dataset sizes and transformer based models, we can learn these tasks to overcome the annotation noise in gaining performance improvements. Pretrained model such as *BERT-base* with high capacity can learn to recognize entities in unstructured text while overcoming the noise added by the automated annotations. Our model performs with a F1-score of 0.80 (**type** metric) on the task of recognition of the named entities. This combination of model and noisy annotations are not adequate for learning exact word tokens belonging to an entity or entity boundaries; thus a low **exact** score was obtained. Combination of federated learning and automated annotation contributes towards achieving our final goal of overcoming the barrier of privacy. This is evident by the substantial performance on the NER task.

Our experiments show a larger federated dataset created by automated annotation can help in converging a model to a good performance while minimizing the external interactions towards a single worker with an isolated data source, given the model's proven ability to learn from a vast number of data points without over-fitting. This ability is evident from the increasing accuracy on the test set as the data volume grows. We also discover in a scenario operated by a mistrusted central operator, despite the model's ability, the final performance can be limited by the numerical precision losses due to intermediate representations necessary for weight transfers in a federated learning system.

With increasing concerns on privacy protection and mass hysteria of public in making sensitive data available for research, federated learning from isolated sources of data is a far less risky approach where it is applicable. This sort of a mechanism can guarantee privacy by being compatible with legal requirements such as ”right to be forgotten” which takes care of the individuals concern on data leakage. When compared with the methods such as de-identification used on unstructured text which possess a risk of breach once the data is released, privacy by isolation is a less risky, practical, comprehensible for the public and adds less responsibility to the data collectors in allowing data for utilization.

Our programmatically annotated dataset carries PII of 5 classes and spreads across a number of demographics due to its source being Wikipedia biography pages. As much as our dataset helps the model to learn entity recognition, it carries noisy annotations as our comparisons show, but increasing size of such noisy data can compensate for its noise as shown by our results. Approaches like snorkel [31] had been embracing noise in annotations when it can be done programatically and then compensated the noise by adding to the volume of inexpensive data. Our approach is the same idea done in a different way. We benefited from the fact that a version of entities were available in the info box and used that to generate noisy inexpensive data annotations, whereas snorkel uses multiple noisy parallel annotation functions and weak-supervision from alternative sources.

5.2 Summary of Contributions

We release our dataset which can be utilised in PII extraction based research. This dataset can be used and shared as it is built from Wikipedia and does not involve any privacy issues. We suggest researchers to use this dataset as a benchmark for comparing new ideas. However, the dataset cannot be used to train fully automated de-identification systems, due to the high level of noise in the annotations. We also release our manual annotation tool which can be used to post process the programmatic annotations from our dataset.

We contribute to the PySyft framework with developments that are crucial for the future in modifying the state-of-the-art models to be executed in federated learning systems. Our modified *BERT-base* language model for remote execution can be utilized in NLP tasks to be trained with federated learning on distributed datasets.

Our contributions as reusable implementations on the remote executable *BERT-base* model are presented under Section B.2 in Appendix B. The repository listed under Section B.1.1 guides through the utilization of our dataset and executing federated learning using the *BERT-base*.

5.3 Future Works

With the availability of resources, we were limited to run our experiments on 16GBs of GPU memory with which we could safely experiment with a model of size as *BERT-base*. In future, by utilizing more resources, pre-trained models of the similar architecture with a higher number of parameters can be trained in these federated settings after appropriate modifications.

Due to the resource limitations and the development time we were unable to experiment on the alternative federated learning settings such as weight averaging. In weight averaging several remote workers train the model on their isolated dataset simultaneously and send the trained weights to the central worker. The central worker averages the weights to update the parameters. This setting would cost multiple amounts of resources as used in centralized or other federated settings we implemented. To run this sort of a system, the PySyft framework also has to be modified with new developments that can manage a parallel training process.

We observed the loss of accuracy and the offset in the value of training loss when models are trained remotely. These phenomenon has to be repeated with controlled experiments and diagnose the proper root cause that leads to this behavior while models are transferred between workers. It would be necessary to relate these representation losses towards the final performance loss in general if they are verified to be existing.

Regarding future work on our dataset, we left several meta data untouched in the biography pages due to our time limitation in creating the dataset and experimentation with the model as an overall task. One such improvement that can be done is taking the section names of sentences in the page and giving a bias towards certain entities to be tagged in that section of the text. For example parent names would be highly probable under the section ”Early life” and spouse names would be more frequent in the section ”Personal Life”. Also ”Early life” can mention education institutes as relevant to personal educational history, an annotation of the education entity in this section would be of higher confidence. These details can be utilized in the automated annotation to increase the accuracy of the annotations.

We currently created the dataset in a central location and distributed it in creating federated learning scenarios. Since our automated annotation method relied only on the data entries itself and used NLP libraries in annotation, this method can be extended to similar data sources without a limitation to the process of creating the dataset. But in a practical scenario, the data source still has to be curated remotely in utilizing automated annotation and federated learning for isolated protection of text resources. Tools for such tasks are not mature in the current framework. It would be highly beneficial to have tools for text manipulation connected in the federated learning system. Then we can have raw data under

every remote worker annotated remotely before using their data for training. It is a quite challenging task to be carried out remotely, if we were to gain the same control as the remote execution of the model due to the heterogeneous nature of the tasks involved. But this remote training can benefit in utilizing many data sources where the data is present in the form of text and are not allowed to leave the premises.

Appendix A

Implementation Details

A.1 Dataset Creation

This section lists few important code listings and algorithms used in the dataset creation.

A.1.1 Info Box Scraping

Listing A.1 present the Python function developed for scraping an info box from a Wikipedia biography page which is given by the Algorithm 1 under Info box scraping topic in Section 3.1.2. This function returns a dictionary with row names as the keys and the text contents of the rows as their entries.

```
class InfoCard:

    '''accepts a beautiful soup html table from a wikipedia \
       page and scrapes the table to a dictionary'''

    def __init__(self, page_content):
        self.info_table = {}
        if not page_content.table:
            raise Exception('page content is incomplete')
        for table_entry in page_content.table_entry_list:
            try:
                left_col = table_entry.find('th', attrs={'scope': 'row'
                        })
                right_col = left_col.next_sibling
                self.info_table[self._row_filter(left_col.text)] = \
                    self._get_text_parts(self._get_kids(right_col))
            except:
                pass
        self.info_table_unfiltered = self.info_table
        self.info_table = self._filter_info_scrapes(self.info_table)
        print 'info card is scraped successfully'
        logging.debug(self.info_table)

    def _row_filter(self, text):
        return re.sub('\xa0', ' ', text)

    def _get_kids(self, html_mother):
        kid_list = []
        try:
            kids = html_mother.children
            for kid in kids:
                kid_list.append(self._get_kids(kid))
        except:
            return html_mother
        return kid_list

    def _get_text_parts(self, text_lists):
        text_parts = []
        if type(text_lists) == list:
            for element in text_lists:
                text_parts.extend(self._get_text_parts(element))
        else:
            text_parts.append(text_lists)
        return text_parts

    def _filter_info_scrapes(self, scape_dict):
        mask_dict = {}
        for (key, val) in scape_dict.items():
            mask_dict[key] = []
            for (i, element) in enumerate(scape_dict[key]):
                name = ''
                lst = [' ' + nm[0] for nm in [part.split('(')
                       for part in element.split(' ')] if nm[0] != '']
                element = ''.join(lst).strip()
                fnd = [c in element for c in u'[]\n\xa0']
                if not True in fnd:
                    mask_dict[key].append(element)
        return mask_dict
```

Listing A.1: Info box scraping function

A.1.2 Entity Tagging

A simplified version of the algorithm given by Algorithm 4 for Fuzzy string matching under Entity tagging topic in Section 3.1.2 is elaborated here. We are utilizing the implementation published by Mike Axiak [1]. Listing A.2 lists a simplified implemetation of the fuzzy matching. Listing A.4 and Listing A.5 show an internal ngram dictionary representation and final matching scores for a list of strings when matched with one string.

```
from math import ceil
from collections import Counter

def pad_len(l, n):
    return n * ceil(l / n) - l

def pad(st, n):
    pn = pad_len(len(st), n)
    return st + ''.join('-' for i in range(pn))

def nGramCount(st):
    dict_list = []
    for nz in range(2, 4 + 1):
        ngram_dict = Counter()
        stpe = pad(st, nz)
        for idx in range(len(stpe) - nz + 1):
            ngram_dict[stpe[idx:idx + nz]] += 1
        dict_list.append(ngram_dict)
    return dict_list

def fuzzyMatchingScore(string_list: list , candidate: str):
    norm_dict = {}
    for st in string_list:
        norm_list = []
        for dic1 in nGramCount(st):
            norm = 0
            dim = 0
            for dic2 in nGramCount(candidate):
                for (key, val) in dic1.items():
                    dim += 1
                    if key in dic2.keys():
                        norm += val * dic2[key]
            norm_list.append(norm / dim)
            norm_dict[st] = sorted(norm_list)[-1]
    return norm_dict
```

Listing A.2: Fuzzy string matching function

```
string_list = ['canadian', 'ithiopian', 'somalian', 'scandinavian']
candidate = 'vanadium'

norm_dict = fuzzyMatchingScore(string_list, candidate)
```

Listing A.3: example usage of fuzzy string matching

A prototype entity tagging function implemented in Python is given by the Listing A.6

[1] https://github.com/axiak/fuzzyset/

```
{'va': 1, 'an': 1, 'na': 1, 'ad': 1, 'di': 1, 'iu': 1, 'um': 1}
{'van': 1, 'ana': 1, 'nad': 1, 'adi': 1, 'diu': 1, 'ium': 1, 'um-': 1}
{'vana': 1, 'anad': 1, 'nadi': 1, 'adiu': 1, 'dium': 1}

{'an': 2, 'ca': 1, 'na': 1, 'ad': 1, 'di': 1, 'ia': 1}
{'can': 1, 'ana': 1, 'nad': 1, 'adi': 1, 'dia': 1, 'ian': 1, 'an-': 1}
{'cana': 1, 'anad': 1, 'nadi': 1, 'adia': 1, 'dian': 1}

{'it': 1, 'th': 1, 'hi': 1, 'io': 1, 'op': 1, 'pi': 1, 'ia': 1, 'an': 1, 'n-': 1}
{'ith': 1, 'thi': 1, 'hio': 1, 'iop': 1, 'opi': 1, 'pia': 1, 'ian': 1}
{'ithi': 1, 'thio': 1, 'hiop': 1, 'iopi': 1, 'opia': 1, 'pian': 1, 'ian-': 1, 'an--': 1, 'n---': 1}

{'so': 1, 'om': 1, 'ma': 1, 'al': 1, 'li': 1, 'ia': 1, 'an': 1}
{'som': 1, 'oma': 1, 'mal': 1, 'ali': 1, 'lia': 1, 'ian': 1, 'an-': 1}
{'soma': 1, 'omal': 1, 'mali': 1, 'alia': 1, 'lian': 1}

{'an': 2, 'sc': 1, 'ca': 1, 'nd': 1, 'di': 1, 'in': 1, 'na': 1, 'av': 1, 'vi': 1, 'ia': 1}
{'sca': 1, 'can': 1, 'and': 1, 'ndi': 1, 'din': 1, 'ina': 1, 'nav': 1, 'avi': 1, 'via': 1, 'ian': 1}
{'scan': 1, 'cand': 1, 'andi': 1, 'ndin': 1, 'dina': 1, 'inav': 1, 'navi': 1, 'avia': 1, 'vian': 1}
```

Listing A.4: Internal representation of the ngram dictionaries

```
{'canadian': 0.28,
'ithiopian': 0.037,
'scandinavian': 0.13,
'somalian': 0.048}
```

Listing A.5: result for the given example

```
import fuzzyset
def _tag_parents(self):
    logging.debug('------PARENTS NAMES------')
    parent_entities = []
    if not 'PARENTS' in self.entity_dict:
        return
    parent_names = [entity for entity_list in self.entity_dict['PARENTS'
                    ] for entity in entity_list['entity_list']]
    for entity in self.doc.ents:
        if entity.label_ == 'PERSON':
            logging.debug(entity.text, entity.label_, entity.start,
                          entity.end)
            parent_entities.append((entity.text, entity.label_,
                                    entity.start, entity.end))
    logging.debug('------PERSON ENTITY MATCHING FOR PARENTS------')
    for parent in parent_entities:
        fz = fuzzyset.FuzzySet(use_levenshtein=False)
        # add all the word tokens into a fuzzy set from that sentence
        for token in parent[0].split():
            fz.add(token)
        for (j, detail) in enumerate(parent_names):  # get a detail
            matched_list = []
            tokens_in_detail = len(detail.split())  # split the detail into word tokens
            for token in detail.split():  # get a token from the detail
                result = fz.get(token)  # get the matching
                 # if the matching confidence is high
                if result and result[0][0] >= 0.5 \
                    # and token character length is high
                    and not len(result[0][1]) / 2 < len(token) / 2:
                    matched_list.append((token, result))
            # if a satisfactory number of tokens match
            if abs(len(matched_list) - tokens_in_detail) < 1 \
                and len(matched_list) != 0:
                annot = (parent[0], 'PR', parent[2:])  # create the annotation
                logging.debug(annot)
                self.tag_set.add(HashableTupleAnnotations(annot))  # add it to the set

    logging.debug('------NOUN CHUNCK MATCHING FOR PARENTS------')
    for noun in self.spacy_noun_chunks:  # ex: ('Joachim Wilhelm Gauck', 0, 4)
        fz = fuzzyset.FuzzySet(use_levenshtein=False)
        noun_tokens = noun[0].split()
        # add all the word tokens from the noun phrase into a fuzzy set
        for token in noun_tokens:
            fz.add(token)
        for (j, detail) in enumerate(parent_names):  # get a one candidate
            matched_list = []
            tokens_in_detail = detail.split()
            if len(noun_tokens) < len(tokens_in_detail):
                continue
            for (i, token) in enumerate(tokens_in_detail):  # get a token from the detail
                result = fz.get(token)  # get the matching
                # if the matching confidence is high
                if result and result[0][0] >= 0.5 \
                    and not len(result[0][1]) < len(token) \
                    and result[0][1][0] == token[0]:
                    matched_list.append((token, result, i))
            # if a satisfactory number of tokens match
            if len(matched_list) != 0 and matched_list[0][2] == 0:
                logging.debug((noun[0], 'PR', noun[1:]))
                self.tag_set.add(HashableTupleAnnotations((noun[0], 'PR'
                                 , noun[1:])))  # create annotatoion and add it to the set
```

Listing A.6: Entity tagging function

A.1.3 Manual Annotator

This section represents an example of manual annotator's usage. A biography entry given by Figure A.1 when auto annotated and loaded with the manual annotator is shown with partially highlighted text as in Figure A.2. Manual annotator can be switched between these annotation highlights and expert's annotations with button "HIGHLIGHT". A manually re-annotated text is given by the Figure A.3.

Figure A.1: Page under annotation

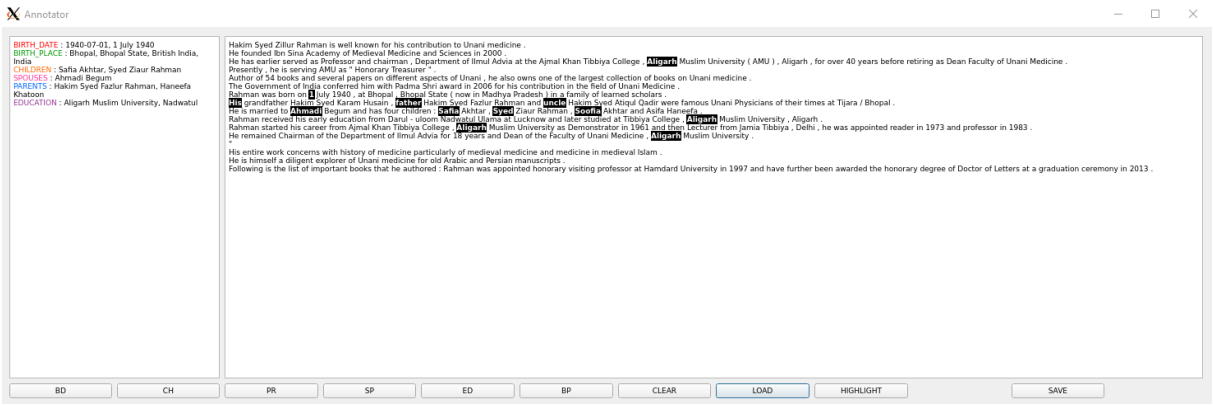

Figure A.2: Annotator UI before manual annotating (only the token of the annotation is highlighted)

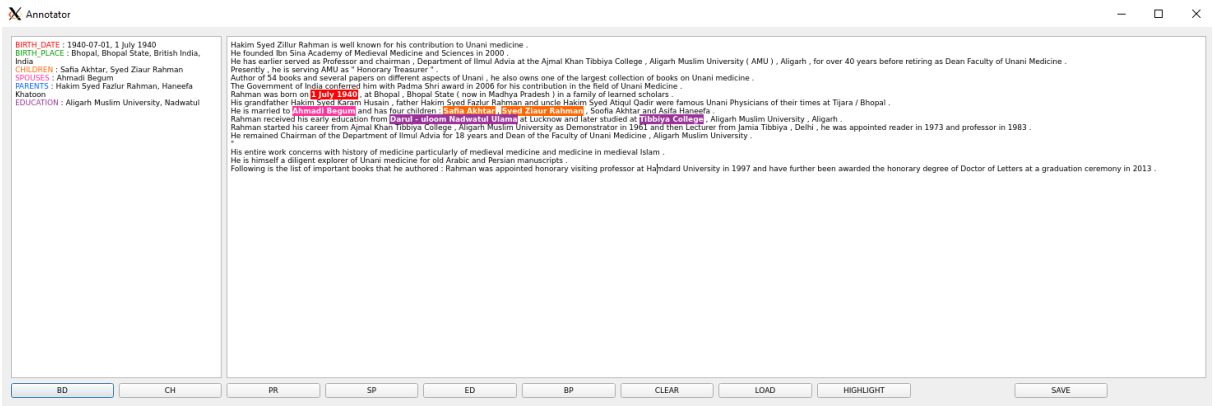

Figure A.3: Annotator UI after manual annotating

A.1.4 Examples of Annotation Ambiguities

An annotation where a parent's name is tagged in a sentence that does not explain the relevant association.

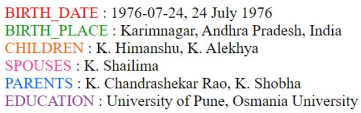

Figure A.4: Info box relevant to the A.5

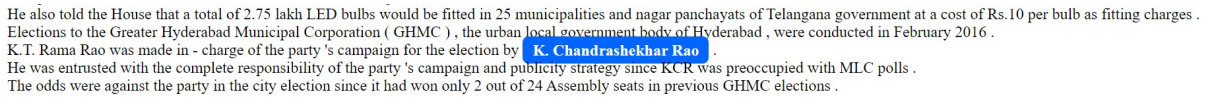

Figure A.5: Ambiguous annotation by the automated annotation

A missed annotation by the the automated annotation but picked up by the manual annotator is highlighted in Figure A.7.

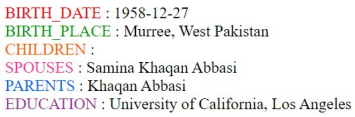

Figure A.6: Info box relevant to the A.7

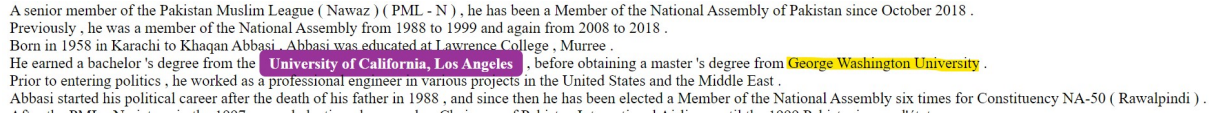

Figure A.7: Ambiguous annotation by the automated annotation

An annotation in a sentence that does not describe the desired meaning is given by figure A.9.

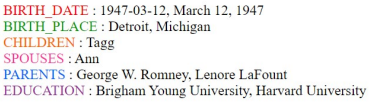

Figure A.8: Info box relevant to the A.9

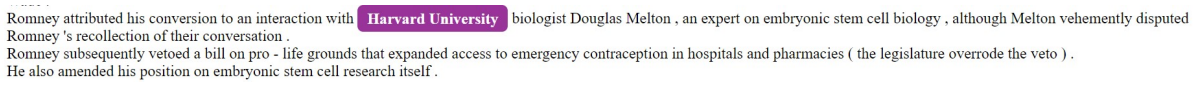

Figure A.9: Ambiguous annotation by the automated annotation

A missed annotations due to entity not being present in the info box but picked up by the manual annotation is highlighted in Figure A.11. It shows another instance of a spurious tagging due to a false matching for the class of ”CHILDREN (CH)”.

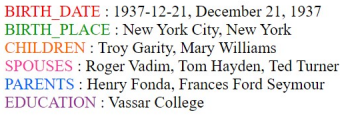

Figure A.10: Info box relevant to the A.11

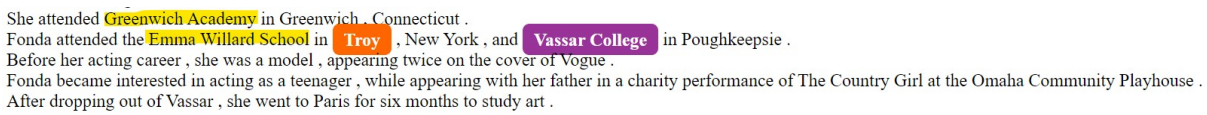

Figure A.11: Ambiguous annotation by the automated annotation

A missed annotation by the the automated annotation due to its alternate utterance in the info box but picked up by the manual annotator is highlighted in the Figure A.13.

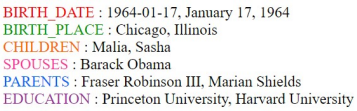

Figure A.12: Info box relevant to the A.13

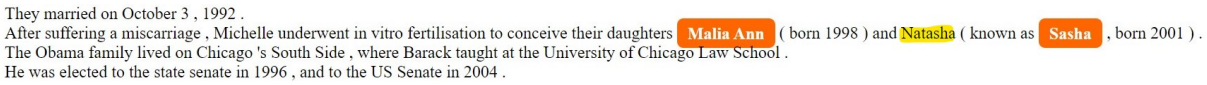

Figure A.13: Ambiguous annotation by the automated annotation

Appendix B

Source Code Contributions

Implementations of our contributions are made available by public repositories. This chapter present the developments available to the public.

B.1 Dataset Creation

B.1.1 Dataset Creation Pipeline

Our dataset creation involves few time consuming processes due to the large number of entries in wikipedia. We release a working sample of all the processes with a smaller number of entries. This is given in the form of a Jupyter notebook. Notebook details the processes under sections and at the end we present how to use our final dataset which is hosted in the Internet Archive[1] at the moment. It can be found in the repository: `https://github.com/ratmcu/wikipiifed`.

B.1.2 Manual Annotator

For much accurate tagging we release our manual annotator application along with the details to add more manual entries to the dataset which we believe can be contributed by the community. This annotator application is release by the repository: `https://github.com/ratmcu/wiki_annotator_ui`

B.2 Federated Learning

Our implementation of the *BERT-base* model for remote operation and the training set up utilizing it are release by the repository `https://github.com/ratmcu/wikipiifed`. A Jupyter notebook guides throughout the developments.

[1] `https://archive.org`

Reference

[1] EU. 2016. REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/EC (general data protection regulation). Retrieved December 26, 2018 from. In: 2018 (*see pp. 1, 10*).

[2] Antonio Sanfilippo and Victor Poznaundefinedski. The Acquisition of Lexical Knowledge from Combined Machine-Readable Dictionary Sources. In: *Proceedings of the Third Conference on Applied Natural Language Processing*. ANLC '92. Trento, Italy: Association for Computational Linguistics, 1992, pp. 80–87 (*see p. 1*).

[3] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. In: *OpenAI Blog*. Vol. 1. 8. 2019, p. 9 (*see pp. 1, 21, 23*).

[4] Wendy W Chapman, Prakash M Nadkarni, Lynette Hirschman, Leonard W D'Avolio, Guergana K Savova, and Ozlem Uzuner. Overcoming barriers to NLP for clinical text: the role of shared tasks and the need for additional creative solutions. In: *Journal of the American Medical Informatics Association*. Vol. 18. 5. 2011, pp. 540–543 (*see p. 2*).

[5] David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. Deep Learning is Robust to Massive Label Noise. In: 2017. arXiv: 1705.10694 [cs.LG] (*see p. 2*).

[6] Trond Linjordet and Krisztian Balog. Impact of Training Dataset Size on Neural Answer Selection Models. In: *European Conference on Information Retrieval*. Springer. 2019, pp. 828–835 (*see p. 2*).

[7] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. In: 2019. arXiv: 1907.11692 (*see p. 2*).

[8] Yaru Hao, Li Dong, Furu Wei, and Ke Xu. Visualizing and Understanding the Effectiveness of BERT. In: 2019. arXiv: 1908.05620 (*see p. 2*).

[9] Stan Matwin, Alexandre Kouznetsov, Diana Inkpen, Oana Frunza, and Peter O'Blenis. A new algorithm for reducing the workload of experts in performing systematic reviews. In: *Journal of the American Medical Informatics Association*. Vol. 17. 4. 2010, pp. 446–453 (*see p. 4*).

[10] Kathleen C Fraser, Isar Nejadgholi, Berry De Bruijn, Muqun Li, Astha LaPlante, and Khaldoun Zine El Abidine. Extracting UMLS Concepts from Medical Text Using General and Domain-Specific Deep Learning Models. In: 2019. arXiv: 1910.01274 (*see p. 4*).

[11] Arvind Narayanan and Vitaly Shmatikov. How to break anonymity of the netflix prize dataset. In: *CoRR*. Vol. abs/cs/0610105. 2006. arXiv: cs/0610105 (*see pp. 4, 12*).

[12] Theo Ryffel, Andrew Trask, Morten Dahl, Bobby Wagner, Jason Mancuso, Daniel Rueckert, and Jonathan Passerat-Palmbach. A generic framework for privacy preserving deep learning. In: 2018. arXiv: 1811.04017 (*see pp. 5, 26, 41*).

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: *CoRR*. Vol. abs/1810.04805. 2018. arXiv: 1810.04805 (*see pp. 6, 17, 23*).

[14] Samuel D Warren and Louis D Brandeis. The right to privacy. In: JSTOR, 1890, pp. 193–220 (*see p. 10*).

[15] Lucas D Introna. Privacy and the computer: why we need privacy in the information society. In: vol. 28. 3. Wiley Online Library, 1997, pp. 259–275 (*see p. 10*).

[16] Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. In: *Official Journal of the European Union L 281, 23/11/ 1995; pp. 31–50.* (*See p. 10*).

[17] Arvind Narayanan and Vitaly Shmatikov. Myths and fallacies of" personally identifiable information". In: vol. 53. 6. ACM New York, NY, USA, 2010, pp. 24–26 (*see p. 11*).

[18] Aris Gkoulalas-Divanis and Grigorios Loukides. Medical data privacy handbook. In: Springer, 2015 (*see p. 11*).

[19] Latanya Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. In: vol. 10. 05. World Scientific, 2002, pp. 571–588 (*see p. 11*).

[20] Latanya Sweeney. Weaving technology and policy together to maintain confidentiality. In: vol. 25. 2-3. SAGE Publications Sage CA: Los Angeles, CA, 1997, pp. 98–110 (*see p. 11*).

[21] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2006, pp. 486–503 (*see p. 12*).

[22] Mike Hintze. Viewing the GDPR through a de-identification lens: a tool for compliance, clarification, and consistency. In: vol. 8. 1. Oxford University Press, 2018, pp. 86–101 (*see p. 12*).

[23] Ishna Neamatullah, Margaret M Douglass, H Lehman Li-wei, Andrew Reisner, Mauricio Villarroel, William J Long, Peter Szolovits, George B Moody, Roger G Mark, and Gari D Clifford. Automated de-identification of free-text medical records. In: vol. 8. 1. Springer, 2008, p. 32 (*see p. 12*).

[24] Clete A Kushida, Deborah A Nichols, Rik Jadrnicek, Ric Miller, James K Walsh, and Kara Griffin. Strategies for de-identification and anonymization of electronic health record data for use in multicenter research studies. In: vol. 50. Suppl. NIH Public Access, 2012, S82 (*see p. 12*).

[25] Stephane M Meystre. De-identification of unstructured clinical data for patient privacy protection. In: *Medical Data Privacy Handbook*. Springer, 2015, pp. 697–716 (*see p. 12*).

[26] Venkatesan T Chakaravarthy, Himanshu Gupta, Prasan Roy, and Mukesh K Mohania. Efficient techniques for document sanitization. In: *Proceedings of the 17th ACM conference on Information and knowledge management*. 2008, pp. 843–852 (*see p. 12*).

[27] Özlem Uzuner, Yuan Luo, and Peter Szolovits. Evaluating the state-of-the-art in automatic de-identification. In: vol. 14. 5. BMJ Group BMA House, Tavistock Square, London, WC1H 9JR, 2007, pp. 550–563 (*see p. 13*).

[28] Kuzman Ganchev, Fernando Pereira, Mark Mandel, Steven Carroll, and Peter White. Semi-automated named entity annotation. In: *Proceedings of the linguistic annotation workshop*. 2007, pp. 53–56 (*see p. 14*).

[29] Hospice Houngbo and Robert E Mercer. An automated method to build a corpus of rhetorically-classified sentences in biomedical texts. In: *Proceedings of the first workshop on argumentation mining*. 2014, pp. 19–23 (*see p. 14*).

[30] Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R Curran. Learning multilingual named entity recognition from Wikipedia. In: *Artificial Intelligence*. Vol. 194. Elsevier, 2013, pp. 151–175 (*see pp. 14, 46*).

[31] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. In: *The VLDB Journal*. Vol. 29. 2. Springer, 2020, pp. 709–730 (*see pp. 14, 55*).

[32] David Corney, Dyaa Albakour, Miguel Martinez, and Samir Moussa. What do a Million News Articles Look like? In: *Proceedings of the First International Workshop on Recent Trends in News Information Retrieval co-located with 38th European Conference on Information Retrieval (ECIR 2016), Padua, Italy, March 20, 2016*. 2016, pp. 42–47 (*see p. 15*).

[33] Matthew Honnibal and Mark Johnson. An improved non-monotonic transition system for dependency parsing. In: *Proceedings of the 2015 conference on empirical methods in natural language processing*. 2015, pp. 1373–1378 (*see p. 15*).

[34] Jinho D Choi, Joel Tetreault, and Amanda Stent. It depends: Dependency parser comparison using a web-based evaluation tool. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2015, pp. 387–396 (*see p. 15*).

[35] David Nadeau and Satoshi Sekine. A Survey of Named Entity Recognition and Classification. In: *Linguisticae Investigationes*. 2007, pp. 3–26 (*see p. 15*).

[36] Lev Retinov and Dan Roth. Design Challenges and Misconceptions in Named Entity Recognition. In: *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL)*. Boulder, Colorado: Association for Computational Linguistics, 2009, pp. 147–155 (*see p. 16*).

[37] F. Lisa Rau. Method for extracting company names from text. In: US Patent US5287278A. 1992. URL: https://patents.google.com/patent/US5287278A (*see p. 16*).

[38] Appendix C: Named Entity Task Definition (v2.1). In: *Sixth Message Understanding Conference (MUC-6): Proceedings of a Conference Held in Columbia, Maryland, November 6-8, 1995*. 1995. URL: https://www.aclweb.org/anthology/M95-1024 (*see p. 16*).

[39] Gang Luo, Xiaojing huang, Chin-Yew Lin, and Zaiqing Nie. Joint Named Entity Recognition and Disambiguation. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Spain: Association for Computational Linguistics, 2015, pp. 879–888 (*see p. 16*).

[40] Alexandre Passos, Vineet Kumar, and Andrew McCallum. Lexicon Infused Phrase Embeddings for Named Entity Resolution. In: 2014. arXiv: 1404.5367 [cs.CL] (*see p. 16*).

[41] Christoph Goller and Andreas Kuchler. Learning task-dependent distributed representations by backpropagation through structure. In: *Proceedings of International Conference on Neural Networks (ICNN'96)*. Vol. 1. IEEE. 1996, pp. 347–352 (*see p. 16*).

[42] James Hammerton. Named Entity Recognition with Long Short-Term Memory. In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. CONLL '03. Edmonton, Canada: Association for Computational Linguistics, 2003, pp. 172–175 (*see p. 16*).

[43] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2013, pp. 6645–6649 (*see p. 16*).

[44] Jason PC Chiu and Eric Nichols. Named entity recognition with bidirectional LSTM-CNNs. In: *Transactions of the Association for Computational Linguistics*. Vol. 4. MIT Press, 2016, pp. 357–370 (*see p. 16*).

[45] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In: *Advances in neural information processing systems*. 2015, pp. 3079–3087 (*see p. 17*).

[46] Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. Semi-supervised sequence tagging with bidirectional language models. In: 2017. arXiv: 1705.00108 (*see p. 17*).

[47] Nick Jardine and Cornelis Joost van Rijsbergen. The use of hierarchic clustering in information retrieval. In: *Information storage and retrieval*. Vol. 7. 5. Elsevier, 1971, pp. 217–240 (*see p. 17*).

[48] Leon Derczynski. Complementarity, F-score, and NLP Evaluation. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. 2016, pp. 261–266 (*see p. 18*).

[49] Cornelis Joost Van Rijsbergen. Foundation of evaluation. In: *Journal of documentation*. MCB UP Ltd, 1974 (*see p. 18*).

[50] Satoshi Sekine and Hitoshi Isahara. IREX project overview. In: *Proceedings of the IREX Workshop*. 1999, pp. 7–12 (*see p. 18*).

[51] Erik F Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In: *Proceedings of the 7th Conference on Natural Language Learning, Edmonton, Canada*. 2003, pp. 142–147 (*see pp. 18, 46*).

[52] Richard Tzong-Han Tsai, Shih-Hung Wu, Wen-Chi Chou, Yu-Chun Lin, Ding He, Jieh Hsiang, Ting-Yi Sung, and Wen-Lian Hsu. Various criteria in the evaluation of

biomedical named entity recognition. In: *BMC bioinformatics*. Vol. 7. 1. Springer, 2006, p. 92 (*see p. 18*).

[53] Nancy Chinchor and Beth Sundheim. MUC-5 Evaluation Metrics. In: *Fifth Message Understanding Conference (MUC-5): Proceedings of a Conference Held in Baltimore, Maryland, August 25-27, 1993*. 1993 (*see p. 18*).

[54] Isabel Segura Bedmar, Paloma Martînez, and Marîa Herrero Zazo. Evaluation Metrics for the task 9.1: Recognition and Classification of Pharmacological substances (ddiextraction 2013). In: 2013 (*see p. 18*).

[55] Isar Nejadgholi, Kathleen C. Fraser, and Berry De Bruijn. Extensive Error Analysis and a Learning-Based Evaluation of Medical Entity Recognition Systems to Approximate User Experience. In: 2020. arXiv: 2006.05281 [cs.CL] (*see p. 19*).

[56] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In: A field guide to dynamical recurrent neural networks. IEEE Press, 2001 (*see p. 20*).

[57] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In: 2018. arXiv: 1803.02155 (*see p. 20*).

[58] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, et al. HuggingFace's Transformers: State-of-the-art Natural Language Processing. In: 2019. arXiv: 1910.03771 (*see pp. 20, 22*).

[59] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In: 2017. arXiv: 1706.03762 [cs.LG] (*see p. 21*).

[60] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. In: 2018 (*see pp. 21, 23*).

[61] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. In: 2019. arXiv: 1901.02860 [cs.LG] (*see pp. 21, 23*).

[62] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. In: 2016. arXiv: 1609.08144 (*see p. 21*).

[63] Jakub Konevĉnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtarik. Federated optimization: Distributed machine learning for on-device intelligence. In: 2016. arXiv: 1610.02527 [cs.LG] (*see p. 23*).

[64] Ji Liu and Stephen J Wright. Asynchronous stochastic coordinate descent: Parallelism and convergence properties. In: *SIAM Journal on Optimization*. 2015, pp. 351–376 (*see p. 24*).

[65] Ji Liu, Steve Wright, Christopher Ré, Victor Bittorf, and Srikrishna Sridhar. An asynchronous parallel stochastic coordinate descent algorithm. In: *International Conference on Machine Learning*. 2014, pp. 469–477 (*see p. 24*).

[66] Peter Richtárik and Martin Takáč. Distributed coordinate descent method for learning with big data. In: *The Journal of Machine Learning Research*. Vol. 17. 1. JMLR, 2016, pp. 2657–2681 (*see p. 24*).

[67] Olivier Fercoq, Zheng Qu, Peter Richtárik, and Martin Takáč. Fast distributed coordinate descent for non-strongly convex losses. In: *2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE. 2014, pp. 1–6 (*see p. 24*).

[68] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. In: 2018. arXiv: `1802.08232 [cs.LG]` (*see p. 24*).

[69] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In: *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2017, pp. 19–38 (*see p. 24*).

[70] Dan Bogdanov, Sven Laur, and Jan Willemson. Sharemind: A framework for fast privacy-preserving computations. In: *European Symposium on Research in Computer Security*. Springer. 2008, pp. 192–206 (*see p. 24*).

[71] Cynthia Dwork. Differential Privacy: A Survey of Results. In: *Proceedings of the 5th International Conference on Theory and Applications of Models of Computation*. TAMC'08. Xian, China: Springer-Verlag, 2008, pp. 1–19 (*see p. 24*).

[72] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep Learning with Differential Privacy. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS '16. Vienna, Austria: Association for Computing Machinery, 2016, pp. 308–318 (*see p. 24*).

[73] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In: *Annual Cryptology Conference*. Springer. 2012, pp. 643–662 (*see p. 24*).

[74] Yoshinori Aono, Takuya Hayashi, Le Trieu Phong, and Lihua Wang. Scalable and secure logistic regression via homomorphic encryption. In: *Proceedings of the Sixth*

ACM Conference on Data and Application Security and Privacy. 2016, pp. 142–144 (*see p. 25*).

[75] Miran Kim, Yongsoo Song, Shuang Wang, Yuhou Xia, and Xiaoqian Jiang. Secure logistic regression based on homomorphic encryption: Design and evaluation. In: *JMIR medical informatics*. Vol. 6. 2. JMIR Publications Inc., Toronto, Canada, 2018, e19 (*see p. 25*).

[76] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. In: vol. 10. 2. ACM New York, NY, USA, 2019, pp. 1–19 (*see p. 25*).

[77] Yann Collet. Lz4: Extremely fast compression algorithm. In: 2013. URL: `https://lz4.github.io/lz4/` (*see p. 26*).