Multilingual De-Identification of Electronic Health Records

Sanchez, I. (Nacho)

Award date:
2020

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics and Computer Science
Information Systems W&I Research Group

Multilingual De-Identification of Electronic Health Records

Master Thesis

Ignacio Sánchez

Committee Members:

Sebastian Menke (Savana Médica)
Renata Medeiros de Carvalho
Joaquin Vanschoren

Eindhoven, October 2020

Abstract

Electronic Health Records (EHRs) are digital pieces of information that store patients' clinical data. EHRs may contain a patient's medical history, diagnoses, medications, treatment plans, and laboratory results. Thus, EHRs represent a rich source of unstructured individual health data that accumulates in hospitals worldwide. The aggregated analysis of the content of millions of EHRs using modern NLP techniques can give insights into diseases that were not achievable with traditional methods. However, EHRs often contain Personal Health Information (PHI) and the protection of such data is crucial. Therefore, authorities demand that anyone exploiting this information must obey the respective regulations regarding data privacy. For the purpose of accelerating health science, companies such as Savana Médica are striving to democratize the clinical value enclosed within EHRs. Savana Médica makes use of AI techniques to extract this valuable information from EHRs. In order to comply with the respective data protection regulations, Savana Médica has developed a set of in-house tools to satisfy those requirements. Their de-identification approach, developed for the Spanish market, needs to be adapted to other languages due to Savana Médica's expansion to other countries. The development of de-identification tools for medical free text is still a challenging field of NLP research. Only few software exist that reach a satisfactory de-identification and most of them focus on the English language. There have been a set of open challenges in the past years on de-identification that have been pushing the innovation in this field. Nevertheless, for many languages such tools do not exist yet and there is a need to develop pipelines that enable to de-identify free text in other languages quickly and of sufficient quality. In this study, we implemented a pipeline for the rapid creation of de-identification tools in different languages using an already existing in-house tool as a basis for development without the need to access privacy protected EHRs in those languages.

Contents

List of Figures viii

List of Tables x

1 Introduction **1**

1.1 Background Information 1

1.1.1 Electronic Health Records 1

1.1.2 Savana Médica 2

1.1.3 Personal Health Information 3

1.2 Motivation & Objectives 5

1.3 Document Overview 6

2 Literature Review **7**

2.1 Rule-based System 7

2.2 Machine Learning System 8

2.3 Deep Learning System 9

2.3.1 Word Embeddings 9

2.3.2 Neural Networks 10

2.4 Hybrid System 11

3 Tools & Methodology **12**

3.1 Technical Environment . . . 12

3.1.1 Cloud Computing . . . 12

3.1.2 Data Sources . . . 13

3.2 EHRs Intake . . . 13

3.3 EHRs Translation . . . 14

3.3.1 Pre-Processing . . . 14

3.3.2 Placeholder . . . 16

3.3.3 Machine Translation . . . 16

3.3.4 Post-Processing . . . 17

3.4 Hybrid System . . . 18

3.4.1 Rule-based . . . 19

3.4.2 Deep Learning . . . 20

4 Results & Discussion **26**

4.1 EHRs Intake . . . 26

4.2 EHRs Translation . . . 27

4.2.1 Pre-Processing . . . 27

4.2.2 Machine Translation . . . 28

4.2.3 Post-Processing . . . 29

4.3 Hybrid System . . . 30

4.3.1 Rule-based . . . 30

4.3.2 Deep Learning . . . 30

5 Conclusions **36**

5.1 Accomplishments & Limitations . . . 36

5.2 Future Work . . . 38

Bibliography **39**

List of Figures

4.1 Summary statistics for 10.000 extracted EHRs 28
4.2 Rule-based unit testing for French emails 30
4.3 Training losses of spaCy's model (language) 32
4.4 Training losses of in-house model (language) 33
4.5 Training losses of spaCy's model (dropout) 35

List of Tables

1.1 PHI types by the HIPAA 4

2.1 Regular expressions examples 8

3.1 Transition sequence example from Lamplet et al. [17] 22

4.1 Size (in MB) and Performance of spaCy's pre-trained language models in French (with standard configuration and 1.000 records) 31

4.2 Performance of in-house model (kernel size) in French 31

4.3 Performance of spaCy's model (language) 32

4.4 Performance of in-house model (language) 33

4.5 Performance of spaCy's model (gold vs silver) 33

4.6 Performance of spaCy's model (letter case) 34

4.7 Performance and Training Time of spaCy's model (data augmentation) . . . 34

4.8 Performance of spaCy's model (data augmentation) 34

4.9 Size (in MB) and Performance of both architectures in French 35

Chapter 1

Introduction

1.1 Background Information

1.1.1 Electronic Health Records

An Electronic Health Record (EHR) is a digital version of a patient's medical history, containing all patient-related data, thereby not only allowing archiving, but also the flow of patient information between authorized healthcare providers and personnel. EHR systems are built to provide a broader view of a patient's care, thus, healthcare specialists can make evidence-based decisions accounting for clinicians' contributions to the patient's history [10]. EHRs include any clinical aspect derived from the patient, not only the doctor's appointment annotations.

To be more specific, EHRs contain past medical histories of patients, including demographic information, contact details, administrative notes and billing data. On top of that, prescriptions as well as allergies and dates of vaccination can be accessed. Furthermore, also laboratory test results and radiology images are often included. Having a patient's comprehensive medical history available in this format allows for a safer and higher quality of patient's care by improving the decision made by doctors. In addition, with respect to the traditional house-keeping of patient data, EHR systems are cost effective and the patient's sensitive information is better protected, which eases privacy concerns [21].

Traditionally, EHR systems were pre-structured so that only certain information was allowed in certain fields. Nevertheless, this practice was more and more discouraged since the complexity of clinical reality cannot be contemplated by a set of drop-down menus. Promoted by a collaboration between Standards Development Organizations (SDOs), a standardization of the EHR format was required for compatibility among healthcare providers utilizing different tools and software.

It was in 1994 that the Health Level Seven (HL7) was accredited by the more than 50 countries represented in the SDOs. HL7 aligns all information systems available in any health domain

by establishing a set of standard formats for each of the categories. In its last version, HL7 defined a series of secured text messages (interactions) to support all healthcare workflows based on the Extensive Markup Language (XML) encoding syntax.

HL7 has been currently adopted by more than 95% of EHR software manufacturers worldwide. This has boosted the adoption of EHR systems among hospitals and EHRs are present in almost any hospital in developed countries. The heavy use of these systems generate an ever-growing amount of valuable information, however, only a minor portion is leveraged today and mainly in the form of scientific research. The main reason that prevented the large-scale use of EHRs' content is that the vast amount of information is presented in the form of free text [12].While a single patient history for a certain disease is not sufficient to drive to a conclusion, the aggregation of many can gain insights to support evidence-based decisions in healthcare.

Only in the Spanish healthcare system, every ten minutes tens of thousands of EHRs are created [11]. This reveals the immense growth of available information which until recently could not be extracted at large scale due to strong limitations in computational power as well as the lack of machine learning techniques able to exploit the valuable information from free text. Nowadays, improvements in Natural Language Processing (NLP) together with the availability of computational power opened doors to a way of dealing with such data that was not possible before. NLP is a branch of artificial intelligence that enables computers to understand and manipulate human language. NLP combines many disciplines, the most important of which are computer science and computational linguistics, filling up the gap between human communication and computer understanding [26].

1.1.2 Savana Médica

A company that creates Real World Evidence (RWE, [7]) based on the information extracted from the free text of millions of EHRs is Savana Médica. Savana Médica started in 2013 as a platform for clinical decision support, based on real-time dynamic exploitation of all the information contained in EHRs of a partnering hospital. Today, Savana Médica offers a wide range of products to clients such as:

Savana Manager

Savana Médica provides the hospital with a user interface (Savana Manager[1]) with which users can aggregate patient data according to their interests. Query results can be visualized and descriptive statistics are performed ad-hoc.

Savana Research

Going one step further in terms of functionalities, Savana Médica automates the extraction and analysis of variables for clinical research and then allows for inference of previously unknown correlations (Savana Research[2]).

[1] https://www.savanamed.com/products/savana-manager/
[2] https://www.savanamed.com/products/savana-research/

Savana Consult

At an operational level, Savana Médica delivers real-time recommendations of scientific evidences to clinicians while filling out the EHR form (Savana Consult[3]).

Savana Predict

With the help of Big Data techniques, Savana Médica generates personalized predictions for adjusting each patient's risk and makes them actionable from any device in real time (Savana Predict[4]).

To be able to offer those services, Savana Médica's NLP pipeline needs to be able to detect a huge variety of clinical concepts and all linguistic features that appear along-side with their mentioning in the free text of EHRs. This is a very complex task because EHRs, amongst others, are unstructured, often incomplete, contain lexical and semantic errors, and many acronyms. Thus, a wide range of specific applications "read" and extract information of interest from EHRs, some of which are:

- Entity detection module: detection of clinical concepts in the free text.
- Disambiguation module: links the detected entity text a unique clinical concept.
- Section detection module: identifies the section to which a paragraph belongs to, i.e. "Background", "Diagnosis" or "Treatment", because not always is the text written below the respective header.
- Negation/Speculation module: detects whether a clinical concept is negated or if it is speculated on by analyzing the concepts' surrounding textual context (such as negation or speculation adverbs: "not", "maybe", "probably"...).
- Temporality module: relates detected clinical concepts to a mentioning of time in the free text.

Apart from the above-mentioned difficulties in extracting clinically relevant information from EHRs, there exist many additional factors that make the exploitation of such data not straight-forward. One of the most important challenges that has to assessed when working with patient data is the sensitivity of personal data, which is subject to data privacy laws.

1.1.3 Personal Health Information

Protecting private data is of uppermost importance to Savana Médica. All the procedures undertaken by Savana Médica are subject to General Data Protection Regulation (GDPR) [13] in Europe and to the Health Insurance Portability and Accountability Act (HIPAA) [3] in the US.

[3] https://www.savanamed.com/products/savana-consult/
[4] https://www.savanamed.com/products/savana-predict/

Table 1.1: PHI types by the HIPAA

No.	PHI Type
1	Names
2	Geographic subdivisions smaller than a state
3	Dates
4	Telephone Numbers
5	Vehicle Identifiers
6	Fax Numbers
7	Device Identifiers and Serial Numbers
8	Emails
9	URLs
10	Social Security Numbers
11	Medical Record Numbers
12	IP Addresses
13	Biometric Identifiers
14	Health Plan Beneficiary Numbers
15	Full-face photographic images and any comparable images
16	Account Numbers
17	Certificate/License Numbers
18	Any other unique identifying number, characteristics or code

GDPR is considered the toughest privacy and security law in the world. It imposes obligations onto corporations anywhere in the world if they are dealing with European personal data. GDPR leverages security by imposing heavy fines to those violating it. In the specific context of Savana Médica, recital 26 and article 32 are key to understand the limitations of the technical design.

Recital 26 states that GDPR should apply to any information concerning an identifiable natural person. For the case of pseudonymized personal data, it states that if it could be attributed to a natural person by the use of additional information, then it is considered information of an identifiable person.

Article 32 of the GDPR requires Data Controllers (hospitals) and Data Processors (Savana Médica) implement technical and organizational measures that ensure a level of data security appropriate for the level of risk presented by processing personal data.

In the US market, on the other hand, Savana Médica is restricted by the HIPAA. The HIPAA states that anyone interested in using data from EHRs has to strip the records of any Personal Health Information (PHI), so that it is de-identified. HIPAA allows for two different methods:

First method is called “Expert Determination”, in which an expert certifies that a person cannot be identified based on the information shown in an EHR. This is a labour-intensive approach that would inhibit companies like Savana Médica to get access to EHRs at the scale needed to offer their services. Second option is called “Safe Harbor”, implies that 18 different identifiers need to be properly removed and replaced with random data so that the document can be considered de-identified. The list of the 18 relevant identifiers is presented in 1.1.

In order to comply with both the GDPR and the HIPAA, Savana Médica aims for the pseudonymization of EHRs by removing each of the 18 identifiers that constitute the Personal Health Information of a patient. Pseudomyization is the process in which personally identifiable information is replaced by synthetic placeholders (pseudonym) which make a person less identifiable while still being suitable for data analysis. Pseudonymization allows for re-identification in conjunction with additional information, whereas a full anonymization irreversibly destroys any information that may lead to the re-identification of a person. Therefore, during the initial "integration phase", EHRs are pseudonymized at the hospital site.

Typically an EHR consists of a structured and an unstructured part. The pseudonymization of the structured part of EHRs is straight-forward. PHI is preceded by a specific label which can easily be matched with simple NLP techniques and subsequently, PHI is replaced with artificial identifiers. The hospital keeps the link to original information, which is never shared with Savana Médica. This ensures that key aspects such as synthetic patient IDs remain consistent between different sources of data (e.g. lab results and clinical notes) so a correct overview of the synthetic patient can be transferred to Savana Médica while protecting PHI.

In terms of free text (the non-structured part of an EHR), PHI may be encountered in any part of the text due to the fact that doctors quite often write the patient name or other PHI in their documentation. To detect those is way more difficult, and more sophisticated approaches are needed to ensure that PHI is correctly identified and removed. For the Spanish market, Savana Médica applies a mix of simple detection methods combined with sophisticated modern NLP techniques such as neural networks to detect these types of PHI.

1.2 Motivation & Objectives

Savana Médica's main focus has been the Spanish market, but the company is currently expanding globally, facing the challenge to rapidly scale and adapt. This directly impacts many parts of Savana Médica's pipeline because they need to be adapted from Spanish to new languages and EHR formats.

Currently, the de-identification suite (*de-PHIEHR*) that Savana Médica has developed for the Spanish market is a combination of many simple to complex functions that detect certain PHI classes (structured ones) and neural networks trained on Spanish documents to identify a patient's name or other more sophisticated PHI classes.

The speed with which the company expands to other countries requires to come up with a pipeline that enables to quickly produce such software for the upcoming languages, which complies with the before-mentioned regulations of data protection. Therefore, the main goal of the research presented in this paper deals with the standardization of a pipeline for the rapid production of a new de-identification system in a new language, taking into account the two components of regular expressions and neural networks for named entity recognition.

How can we build a production-ready multilingual pipeline for de-identifying EHRs with low resources?

This research question comes with various adjacent questions to be solved. Since Savana Médica possesses synthetic records only in Spanish, a full pipeline for the translation and generation of synthetic records to other languages is required.

How can we create realistic synthetic EHRs in other languages?

There are several approaches that may be followed but they vary in time-investment as well as costs. Thus, it is key to come up with a highly efficient solution while optimizing the resources (data, time and computational power).

What is the minimum amount of resources needed for building a working pipeline?

At the same time, to optimize this pipeline, a proper assessment of the de-identification system architecture is crucial and will be analyzed and benchmarked in detail.

What is the most efficient de-identification system for each PHI type?

1.3 Document Overview

After the above introduction including background information on the topic, the motivation and objective of this research project, the paper is structured as follows:

Chapter 2: A detailed review is presented on existing literature about the different topics treated in this paper. There are two main points that encompass the whole scope of the project: 1. The determination of an automated EHR de-identification pipeline, considering and comparing all the possibilities. 2. The aspect of multilinguality in terms of adapting any of the de-identification systems to various languages.

Chapter 3: Presentation of the tools utilized for the realization of the project. It gives details about the technical environment in terms of architecture and computational capabilities. Moreover, it describes the methodology followed in each of the steps of the de-identification pipeline and the reasons for each decision made in the process.

Chapter 4: Results derived from executing the different steps of the defined pipeline are explained. From the database extraction to the deep learning model, configurations for every parameter are delivered so that everything can be easily reproducible. It also addresses the discussion of the results.

Chapter 5: Explanation of most relevant conclusions derived from the research and the limitations encountered together with the future work to do for better results.

Chapter 2

Literature Review

With the urge of of processing large amounts of sensitive health data by many research institutions, the automation of the de-identification of records has been a major challenge in the recent years. De-identification is usually framed as a Named Entity Recognition (NER) problem. NER is a sub-task of information extraction in NLP that aims at identifying and tagging named entities (in our case PHI) mentioned in unstructured text [24]. Going from a simplistic to a more complex architecture, de-identification can be addressed through different NLP techniques such as:

2.1 Rule-based System

In a rule-based system, domain experts rely on pattern matching with dictionaries, regular expressions and other patterns [28]. Systems such as the one presented in [30] fully rely on these methods so they do not have the need of labeling data. Regarding to the domain of EHRs, PHI classes such as names or streets can be detected by determining whether a word is preceded with a specific identifier (e.g. Avenue, Boulevard, Dr., Mrs.). If this is not possible, there is the option to use lookup lists that contain terms such as most common names/surnames, street names, and names of medicines. Fuzzy string matching [1] allows for some level of tolerance regarding spelling errors.

Furthermore, the more structured a PHI class is, such as in the case of emails addresses, phones or medical numbers, the easier the identification using regular expressions (regex). A regular expression is a search pattern used for matching one or more characters within a string[1]. In table 2.1 we can see some special characters used within a regular expression that add certain characteristics to the search pattern.

[1] https://techterms.com/definition/regular_expression

Table 2.1: Regular expressions examples

Symbol	Meaning	Example
.	any character	/a.c/ matches abc
*	zero or more repetitions of previous character	/a*/ matches aaa
?	zero or one repetition of the previous character	/a?/ matches a or null
\a	any alphabetic character	\a = [abc...z]
\d	any numeric character	\d = [012...9]

While rule-based systems can reach high precision, their recall is often limited since it is hard to cover all possible spelling variants of some PHI classes. On top of this, these systems struggle with context-dependent terms that could be misidentified as PHI. A clear case are eponymously named diseases such as "Lou Gehring disease", that may lead to the wrong detection of a name and a surname that should not get pseudonymized. Taking into account the high relevance of these clinical terms, de-identifying them would be a huge mistake for any de-identification system and render the interpretation of such data meaningless.

2.2 Machine Learning System

Looking at the limitations of rule-based systems, the main drawback comes from the fact that all is based on manually developed rules. Thus, some studies work on the automation of both the creation and execution of regular expressions [2]. Such systems combine regular expression discovery (RED) algorithms with support vector machine (SVM) classifiers.

Simplifying the combination described above, some researchers opted for a purely machine learning based approach. De-identification can be seen as a classification task and machine learning approaches like conditional random fields (CRFs), decision trees and SVMs have been used for developing de-identification systems [8][22].

Machine learning (ML) is an area of artificial intelligence (AI) that is based on the concept that a computer program is able to learn and adapt to unseen data without human input [29]. For our de-identification use case, machine learning systems encode each word of the text (token) and its surroundings as a set of features that will then label each of the tokens by letting a classification algorithm learn the most suitable configuration. Such features can be, amongst others, linguistic features such as Part-of-Speech tags (syntactic analysis), or orthographic features such as prefix of the token or lemma (dictionary form) of the token.

Nevertheless, the performance of the system heavily depends on the feature engineering as well as on the size of the labeled data set for the supervised learning task. Even with a good data set, rare patterns that are not present in the training set will not be identified as PHI and the recall may drop. Available models trained on specific text corpora usually do not generalize well on other domains, especially on the medical one.

2.3 Deep Learning System

As machine learning techniques in NLP evolve, the most noticeable improvement is due to artificial neural networks (ANNs). The combination of layers of neural networks has led to the emergence of the so-called deep learning (DL) architectures. DL is currently the state-of-the-art method for NER tasks [6]. DL systems present a main advantage over ML: they do not require feature engineering, which tends to be a time-consuming process in ML systems. Advances in both, word embeddings and DL architectures, have created DL approaches that surpass human performance in many areas that were thought to be impossible to be achieved by computers. As a counterpart, DL systems are difficult to interpret and training is more expensive than for ML frameworks.

2.3.1 Word Embeddings

Neural networks take only digits as input so that when dealing with text, each word must be mapped to a number. Until recently, the most used approach was that of bag-of-words (BoW) representation. BoW is consists on the description of a document without order where only the counts of words matter. The vector representation of those counts can then be used to compare similarities between texts. Nevertheless, this method comes with major limitation, it does not encode the underlying meaning of the words. Mikolov et al. [23] generated a vector space where words get a vector representation according to their usage in the text. By doing so, similar words get assigned similar vector values. A classical example to understand this technique is the below one, in which the embedding vector is represented by X:

$$X_{king} - X_{man} + X_{woman} = X_{queen}$$

Each word is usually represented as a multi-dimensional vector. There exist different approaches to word embedding technique, such as the Word2Vec or GloVe [5]. Both approaches are outlined in more detail below with their most used architectures:

- Word2Vec: there are two versions for this word embedding method depending on the predicting output used for training the model:
  - Continuous Bag Of Words (CBOW): this model is trained on predicting a specific word based on the surrounding words (context).
  - Skip-Gram: contrary to CBOW, the model is trained on predicting the surrounding words (context) based on a specific word.

  In both cases, Word2Vec relies on a feed-forward neural network with the task of predicting the next word (known as language modeling), and are based on optimization techniques such as stochastic gradient descent [18].
- Global Vectors (GloVe): in contrast to the context window-based methods (Word2Vec), GloVe focuses on the co-occurrence of words within the entire corpus, i.e. it looks at

how frequent a word X appears in the surroundings of word Y in the totality of texts available for training.

Even though these pre-trained word embeddings mean a major step forward for DL in NLP, they present some limitations in terms of generalization for different domains corpora. To overcome this, recent developments have led to generate context-dependent representations that capture also various features of each word, leading to a more complete representation of each token [20]. In Khin et al. [15], they get state-of-the-art results on de-identificating EHRs by implementing the Embeddings from Language Models (ELMO) developed by Peters et al. [25].

ELMo[2] representations are learned though a Bi-LSTM architecture with a language modeling (predicting next word) task. On the higher-level LSTM, the model encapsulates the context-dependent aspects of the word. On the other hand, the lower-level LSTM looks for the syntactic features of the token.

Adapting NLP to industrial applications, spaCy proposes an alternative for word embeddings that delivers an efficient solution in terms of speed, size and accuracy: bloom embeddings[3]. This novel strategy with subword features is used to enclose large vocabularies in reduced size tables. Contrary to the Bi-LSTM model, spaCy presents an architecture of Convolutional Neural Networks (CNNs), with residual connections and layer normalization, which delivers much better efficiency than the ELMo embeddings with a minimum cost in performance (only 1% off the state-of-the-art in English NER task).

Going back to the research question of this paper, we look for a production-ready pipeline for Savana Médica's de-identification of EHRs, thus, spaCy's word embeddings seem to be more suitable for our case. It is the architecture proposed for the study and it will be replicated in-house with the available tools. Since there is not enough literature on this novel mechanism, the in-house embeddings may show a lower performance after all.

2.3.2 Neural Networks

Even though word embedding techniques already include neural networks within their different frameworks, NER task requires for an additional prediction layer that performs the task of tagging each of the tokens. Similar to what happens with embeddings, Bi-LSTMs are currently the best performing architectures among all, specially the one based on LSTM-CRF [6].

In line with the decision of the embeddings, this research will focus on spaCy's solution for the NER task [27]. Continuing after the residual CNNs, spaCy decides to change the scope of the tagging task: instead of treating the word as the object of interest, it looks for predicting the set of transition sequences to be done with the stack of words in the text. More details on this will be given in 3.

[2] https://allennlp.org/elmo
[3] https://spacy.io/models

2.4 Hybrid System

By looking at the advantages and disadvantages of all proposed systems, there are studies that present a combination of rule-based method with either machine learning or deep learning architectures. This allows for ad-hoc solutions to each of the different PHI, saving time in annotating if rule-based is chosen (e.g. emails, telephone numbers) and surpassing hetereogenity if machine/deep learning is put in place (names and streets).

It was in 2014 that, for the i2b2[4] challenge, Yang et al. [31] and Liu et al. [19], demonstrated accuracies well over 90% when detecting PHI in EHRs with hybrid systems. These were composed by a set of dictionary look-ups, regular expressions and CRFs. However, these frameworks still heavily relied on the time-intensive task of feature engineering that may not be generalizable to other text corpora.

While entire deep learning systems have proven to perform better in the clinical domain for NER task, Savana Médica is interested in maintaining the pipeline as simple as possible so that low-resource languages can be integrated in an efficient manner. To make it possible, we decide to develop a hybrid system composed by a rule-based module (mainly regex) and a deep learning module. Having 18 different types of PHI, each will be treated by one of the modules. As a sample we will address phone numbers, emails, medical numbers and zips with the rule-based module and names/surnames with the deep learning module.

[4] https://techterms.com/definition/regular_expression

Chapter 3

Tools & Methodology

In this chapter, a complete overview of the technical setup of the project is presented in terms of software tools and architecture. Additionally, details about the steps involved in building the *de-PHIEHR* are given.

3.1 Technical Environment

3.1.1 Cloud Computing

Working with sensitive data implies that all processes must occur in a safe environment, thereby ensuring data protection standards. To comply with those standards, Savana Médica performs all its data operations in a private cloud using Amazon Web Services (AWS). AWS is a cloud computing platform that offers hardware and software solutions to users. Users can select from a wide range of options such as selecting the computational power of Amazon's Elastic Compute Cloud (EC2) or make use of specific software such as translation services.

Savana Médica's *EHRead* team, responsible for the development of the *de-PHIEHR*, uses EC2 instances to comply with data protection regulations. For this project, all computational steps in the production of the *de-PHIEHR* were carried out on an EC2 instance with 4 CPU cores and 16 GB of RAM and GPU. To train deep learning models quicker, computational power was increased when needed.

Jupyter Notebook[1] was the integrated development environment (IDE) used to build the entire pipeline, from the data intake to the model evaluation. It is a testing environment that allows to run chunks of code in a convenient manner by separating them in different cells and allowing for an easier debugging and construction of live code and visualizations.

Python was used as interpreter since many NLP libraries applicable for our use case are scripted in that programming language. Moreover, the EC2 instance is linked to an Amazon

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

Elastic File System (EFS) so that code and data files are securely stored in the cloud[2]. With all files in the same system with EFS, the user can apply code directly on the data and interact with it via Jupyter Notebook.

3.1.2 Data Sources

Due to legal issues, Savana Médica cannot make use of EHRs of partnering hospitals which are not targeted for that specific hospital, such as training models for general purposes. Therefore, to develop software tools like the *de-PHIEHR*, *EHRead* has to comply with contractual restrictions and data privacy laws. For this reason, Savana Médica has established a large pool of medical doctors over time that created a data set of synthetic EHRs manually for different medical specialties in Spanish. Using an advanced algorithm to re-combine those synthetic EHRs led to a rich database of Spanish EHRs. Those are stored in dedicated row-oriented DBMS database management system (DBMS) containing one table per fictional hospital. Each of these tables stores a randomly assigned record ID and the corresponding raw text of each EHR.

In addition, Savana Médica applied its NLP pipeline on those records in order to extract e.g. clinical concepts, and stored them in a dedicated column-oriented DBMS. This allows for each of the apparitions (detected relevant clinical concepts) to have a dedicated row. With this database, it is possible to filter records based on e.g.:

- Services: utilities offered by hospitals in the form of medical and surgical assistance, laboratory and pharmaceutical provisions. E.g.: rheumatology, blood test, oncology.
- Terms: set of clinical terminology that is relevant for each of the studies. This could be concepts such as diseases, symptoms or treatments.
- Patient age: current age of the patient based on available information in the EHR.

Given this architecture, it is possible to relate the row-oriented DBMS and the column-oriented DBMS databases in a way that one can filter records by a set of criteria for apparitions and extract the respective raw text from the row-oriented DBMS.

3.2 EHRs Intake

Developing a deep learning model for the detection of PHI requires an annotated set of EHRs for training and testing. For the random extraction of EHRs from the row-oriented DBMS, we scripted a class in Python with a set of methods that allow the user to perform SQL queries to the row-oriented DBMS based on a set of filtering options that work on the column-oriented DBMS. Users can adjust a configuration file (in .json format) which permits them to determine the following:

[2] https://aws.amazon.com/security/

- Filters: in case the user wants to refine the request, four different parameters can be set when looking up the column-oriented DBMs:
  - Services: each of the allocated departments of a fake hospital.
  - Terms: clinical terminology that may refer to, amongst others, diseases, symptoms, treatments, and medicines.
  - Age: the patient age, although stored as days in the column-oriented DBMS, can be filtered by setting the age in years in the configuration file. A conversion is applied in the background.
- Amount of EHRs: number of documents to be extracted in raw format.
- Amount of characters: minimum number of characters per extracted document.
- Filtered/Non-Filtered: percentage indicating the proportion of filtered records (according to the activated filters) vs. non-filtered records.

In addition, the Python class also handles how the EHRs are stored. EHRs are stored in a dictionary format with a unique key (record-id) for each. This dictionary is then serialized to a byte stream object and saved to disk using pickle[3]. This file can easily be load into other operations as input for another chunk of code from a different file.

3.3 EHRs Translation

3.3.1 Pre-Processing

Before performing the translation to other languages, in our case to French and English due to Savana Médica's future interests, it is important to ensure that the input EHRs are in the correct format. Hence, a set of subsequent actions are introduced in the pipeline after the extraction part, containing the following modules:

Language Detection

Among the synthetic records there are also some that were written in Catalan[4]. Therefore, once extracted from the row-oriented DBMS, to be consistent and keep only those in Spanish, a language detection method is applied to all EHRs. In this case, the selected Python library to perform this task is *langdetect*[5].

langdetect is the preferred choice among all available tools in Python for language identification of a text. It is a direct port of the existing Google library written in Java called *language-detection*[6]. It is currently able to detect 55 languages out-of-the-box.

[3] https://docs.python.org/3/library/pickle.html
[4] https://en.wikipedia.org/wiki/Catalan_language
[5] https://pypi.org/project/langdetect/
[6] https://code.google.com/archive/p/language-detection/

Trained with Wikipedia articles, *langdetect* performs with a 99.8% average precision in all languages. With a simplistic Naive-Bayes algorithm working with character n-grams, the problem is solved as a classification one. Nevertheless, it is a non-deterministic algorithm and it requires a seed to be set to provide consistent results.

In the *de-PHIEHR* pipeline, all extracted EHRs go through the detection function of *langdetect* one by one. After that, any record that does not return "Spanish" as detected language is removed from the set so that translation does not present any downstream issues.

Silver Standard

With only Spanish records remaining, there is a last stage of pre-processing before translation. As the name detection will be based in a supervised-learning model, the dataset for training and testing must be labeled. In order to avoid manual annotation at this point, Savana Médica's in-house model for name and street detection in Spanish free-text is used to identify names in extracted synthetic EHRs. The output format of the Spanish name detector is a dictionary with the following structure:

- Text: raw text of the EHR.
- Annotations:
  - Term: word that has been annotated.
  - Initial Offset: position of the first character of the term.
  - Final Offset: position of the last character of the term.
  - Tag: label of the identified PHI type.

To ameliorate the *silver standard* annotation, two modules are added to the name detection pipeline. The first one is responsible for distinguishing between personal names and those referring to locations (streets, avenues, boulevards...), because we want to limit our approach to a specific PHI class. For the creation of this filter, a list of Spanish street suffixes and their abbreviations[7] is added as a CSV and loaded as a flat list in the Python class.

With the list loaded, by using the regular expressions module in Python and taking as a reference the offsets from the dictionary, the previous and following twenty characters of each identified name are checked for perfect matching with any of the terms in the street suffixes. If there happens to be a match, that name is removed from the annotations of the document and the dictionary gets updated.

Moreover, many clinical concepts are named after e.g. individuals. These are known as eponymous and must not be identified as names in the EHRs. A similar approach to that applied for street suffixes is used. Eponymously Named Diseases (ENDs) lists are freely available online[8]. Our Python calls uses this input and compares the text of each annotation against the ENDs list. Again, if a match occurs, the annotation gets updated in the dictionary.

[7] http://www.wikilengua.org/index.php/Lista_de_abreviaturas_de_v%C3%ADas
[8] http://self.gutenberg.org/articles/eng/list_of_eponymously_named_diseases

3.3.2 Placeholder

Lastly, translation presents another potential issue for certain names that may get translated to other languages. To avoid this, a persistent term needs to be substituted in place of the detected names. After some manual inspection with various online translators, *Faustino* is chosen as a persistent term for English, French and Spanish. Taking advantage of XMIs structure, each of the identified names are substituted by *Faustino* and EHRs get saved as individual files only with the raw text.

Furthermore, we generate a dataframe compiling the main characteristics of each record per row. This includes their text length, number of annotations (*Faustino*s), average initial offset of their annotations, their density (measured in annotations per character) and distribution of annotations (in terms of quartiles). This becomes useful when integrated in the form of summary statistics. These help to perform a filtering so that only relevant records get translated. The statistical information computed includes:

- Average number of annotations.
- Average text length.
- Average initial offset of annotations.
- Distribution of annotations:
  - 1^{st} Quartile.
  - 2^{nd} & 3^{rd} Quartile.
  - 4^{th} Quartile.

Considering all computed summary statistics, a last filtering module is added to the pipeline. We are capable of setting thresholds regarding minimum and maximum number of annotations as well as characters length in terms of its aggregated average and standard deviation. Only those filtered EHRs get saved in a new directory that will be called to perform the translation.

3.3.3 Machine Translation

Even though there are several open-source options for translation, the security that must be implicit in all Savana Médica's processes together with the seek for agility lead to more sophisticated approaches. Thus, taking advantage of the available AWS infrastructure at Savana Médica, the paying service for machine translation is picked: SDL Enterprise Translation Server (ETS).

SDL ETS offers automated translation of raw text by using state-of-the-art machine translation. It presents itself as a solution designed for time sensitive projects such as ours. SDL currently delivers the service in different packages depending on the languages and the direction of the translation. This comes as a limitation when translating from Spanish because

most translations to anything else than English uses this one as a bridge. So the pre-processed records are translated firstly to English and then from English to French. This causes mistakes in first translation that get propagated downstream.

Looking for a cost efficient solution and accounting for the budget available for this project, Savana Médica provides with six payed hours of the translation service. Due to this, the code must take advantage of all computational power available. For that end, a queue of records to be translated is defined from which four different threads are fed. All four threads are independent and each EHR gets saved as an independent file with a JSON format. These documents contain the raw text in both the source and target languages.

As explained previously, records must get translated to English first from Spanish so then English can serve as a join for French translation. In principle, both translations must take similar time frames so that available service hours are split in half. Spanish to English translation will be the limiting one in terms of amount of records that can be processed in three hours. With reference to the generated JSON files, for the second translation both previous languages will be conserved when adding the target language raw text. This means that we end up with a unique file per record containing its raw text in Spanish, English and French.

3.3.4 Post-Processing

Translated records are indirectly annotated by having all names as *Faustino*. Nevertheless, if texts are maintained and labeled as such, our model would just learn that specific name when looking for individual's names and would not perform well in realistic cases with other names in the text. Having said that, a module for post-processing the records after translation is introduced. It includes a manual check for non-detected names in the *silver standard* process together with the substitution of all *Faustino* by real names and surnames.

Gold Standard

In order to create a *gold standard* from the *silver standard*, records are manually curated using an annotation tool to ensure that the data for training our model is perfect. This task could be done prior to translation for faster substitution, but since it is a manual task, the fastest way is by only checking those records that can be translated within the three hours scope.

Manual inspection is alleviated by annotating all *Faustino* present in each document so that it is visually faster to locate new names in the text. This is performed record by record and typically checked by at least two reviewers. To ensure even better performance, since the old name detector tended to learn names (not by their context), there are names that are repeatedly missed by the model. Thus, a white-list containing the manually detected names is generated and EHRs are double-checked against it. The result of both processes go through the replacement by *Faustino* which leads to the creation of our *gold standard*.

Synthetic Random Naming

Once the *gold standard* is available, we replace all *Faustinos* with a real name for each of the languages. To do so one needs data sets of names and surnames for each of the respective languages. There are official sources for the three languages that provide this data in separate CSVs. In Spanish, the Spanish National Institute of Statistics[9] provides names and surnames together with their frequency. Similar data sets are available for French[10] and English[11].

Names and surnames, as present in real EHRs, come in many different formats: all letters in capital, or title case, or surname first, or only surnames, or a mix of those, just to mention a few. In order for the synthetic name replacement to be realistic, letter case variation for name substitution will be sampled from the original records. Three different cases will be assessed: title case (e.g. John Doe), uppercase (e.g. JOHN DOE) and lowercase (e.g. john doe). Frequencies of occurrence will be computed from the original names in Spanish on the synthetic EHRs.

Another aspect to consider is the order of name and surnames. Spain for example tends to use both parents surnames, so that the usual composition consist of a name and two surnames. As we start with EHRs in Spanish, this is the present structure in many of the available EHRs. However, in English and French, full names consist of usually only one surname and therefore some adjustments have to be done. Something else to consider is the existence of compound names and surnames.

To deal with all these possible scenarios, rules are applied when replacing *Faustino* with names and surnames. At last, to prevent model over-fitting, names will be used uniquely for each of the substitutions, this will make the testing always zero-shot. All the modules described in this section are implemented in the form of a class with several methods that allow for testing of different configurations of random naming for EHRs corpora generation.

3.4 Hybrid System

Most of the pre- and post-processing of the data are carried out to prepare the training data for the NER for names of the *de-PHIER*. But, as previously explained, a person's names is not the only sensitive PHI in EHRs. With reference to many other PHI classes (e.g. email, phone, or social security number), they have specific characteristics which can be utilized to easily detect those using regular expressions. Therefore, we implement a hybrid system, mixing rule-based identification with a deep learning model for the name detection.

[9] https://www.ine.es/
[10] https://www.data.gouv.fr/
[11] https://www.data.gov.uk/

3.4.1 Rule-based

PHI classes such as phone numbers and emails can easily be detected using regular expressions (regex) that match their standard format by matching a specific pattern (e.g. number of digits, the presence of "@" and ".com"). Nevertheless, there is a main barrier to the standardization of these regex for different languages because formats might differ. For the scope of the project, only French (from France) will be addressed to serve as example for the other languages.

Language-specific RegEx

The most common PHI classes in our EHRs are: emails, telephone numbers, social security numbers, medical identification numbers and ZIP codes. Each of those must be treated separately, and their specific regular expression needs to be manually generated. Since all PHI is written by the physician, orthographic errors and variations are frequent.

After a thorough research, details about PHI that frequently occur in French EHRs were found in French public sources. In addition, there exist web forums where users share their previously developed regex as an open resource. By contrasting both sources and with the help of an online regex generator[12], the regular expressions for PHIs were created. Those regex are bundled in a list of dictionary which can be orchestrated using a Python script.

This structure allows for the combination of regexes for the same PHI class. By doing so, a PHI class such as a telephone number may be detected in case of a landline or cell phone which have different structures. This approach makes debugging straightforward and eases the addition of new regexes when required by the needs of a specific use-case in a hospital.

With all French regexes in place, there is one last point to take into consideration. Based on our experience working with Spanish records, many of these PHI classes come in a numeric format (e.g. amount of blood cells, drug doses). To avoid that any regex confuses those with actual PHI, supported by the medical knowledge shared by the medical doctors at Savana Médica, some conflicts are anticipated so that relevant data is not lost during the de-identification.

Thus, in addition to the list of dictionaries containing the regexes, a "blacklist_" followed by the PHI class name is created for each. This blacklist contains two pieces of information also in the form of a list of dictionaries. Each of the dictionaries contains a key with a medical term together with a numerical value representing the distance in characters from the match. To see an example, this is the case with ZIP codes and leukocytes. Thus, its black list contains "leukocytes" as key and, after manual inspection of leukocytes appearances, we determine that leukocytes appear in a window of maximum 15 characters from the quantity (false ZIP), then, the value is set to "15".

[12] `https://regexr.com/`

Generic Matcher

With only one language in place, the process to run is straightforward with the use of the created regex library. However, we want to make it adaptable to any language and for that a generic matcher class is developed. This class enables to orchestrate a multilingual set-up for each of the respective PHI classes.

Furthermore, some PHI classes require a second check to assure that the identified entity has been properly matched. For example in France, doctors' identification numbers follow the Luhn formula. This algorithm is also used for validating of credit card numbers. It is based on a checksum of all the digits which must coincide with the last figure to be validated. Luhn's checksum is as follows (according to [14]):

"The Luhn algorithm starts by the end of the number, from the last right digit to the first left digit. Multiplying by 2 all digits of even rank. If the double of a digit is equal or superior to 10, replace it by the sum of its digits. Realize the sum s of all digits found. The control digit c is then equal to

$$c = (10 - (s \ mod 10) \ mod 10).$$"

Thus, a trigger is introduced in the regex module that detects such PHI classes so that whenever French text is being de-identified, a double check with the Luhn's algorithm is done for those. A similar approach is followed for the blacklist check (as occurred with ZIP codes and leukocytes). Ultimately, to link this rule-based module to the entire hybrid system, a dictionary format is chosen. The raw text of the EHR is attached to the annotations which contain for each detected entity the offset, label and matched token text.

Unit Testing

To automatically test the integrity of the code, a set of unit tests are put in place for each one of the regexes. Input and desired output are compared based on the values in a dictionary for each PHI class. When the test suite is executed, a validation message is print out so that users can easily track errors.

3.4.2 Deep Learning

For the more sophisticated task of detecting names in free text, rules are not sufficient anymore and a deep learning approach is chosen instead. For that, a Named Entity Recognition architecture needs to be put in place.

SpaCy

The NLP library spaCy[13] is an open-source library for industrial-strength Natural Language Processing. SpaCy offers several out-of-the-box solutions which can be used as a benchmark for the evaluation. It has a set of tools for processing and understanding large volumes of text. For the specific task of NER, which is the one needed for this project, spaCy has recently developed their version 2 of the module. It outperforms their version 1 by more than 6% (81.4 to 86.4) of F-score for their benchmark, while reducing the model size by a third (1GB to 667MB) for their large version.

a) Embedding Layer

SpaCy's NER begins with an embedding layer responsible for generating vectors for each of the words. After transforming the whole document in a set of tokens, each of these tokens is converted to a vector composed by the following components:

$$(\text{norm} — \text{prefix} — \text{suffix} — \text{shape})$$

where,

- norm: normalized form of the string.
- prefix: first three characters of the string.
- suffix: last three characters of the string.
- shape: it gives a representation of the characteristics of the string (capital letters, digits or special characters).

This type of representation allows for the system to function well with unknown words by extrapolating the form if the norm is not found for example. This may be the case for misspelled words and grammar errors. With all four components set for each word, through a hashing mechanism, spaCy generates unique values for each of them. Afterwards, the hashed features are concatenated and get fed forward into a multi-layer perceptron with one hidden layer and the maxout unit. The output of this layer is a 128 dimensional vector per word that accounts for all sub-word features.

b) Trigram-CNN Layer

While the embedding allows to encode every single term in texts, it does so in a context-independent manner. Thus, spaCy turns this vectors into context-sensitive matrices. The whole point behind this is to transform the generated unique vectors into new ones that consider also their surroundings. Therefore, spaCy uses a trigram Convolutional Neural Network (CNN) layer that takes as input a 384 dimensions vector (128 per word). Then,

[13] https://spacy.io/

using a multi-layer perceptron, the vector gets mapped down to a new 128 dimensions vector. Altogether, spaCy uses 4 trigram-CNN layers that draw information from the four surrounding words on either side of the word's vector. To smooth the output from each layer, residual connections are introduced, so that the output of each layer is the sum of the output and the input. This provides the network with a bias towards not making drastic modifications to the initial vector.

c) Attention Layer

Even though it is not an attention mechanism as such, the feature extraction done over the state vectors can be understood the same way. Instead of summarizing the inputs (set of word vectors of a document), spaCy manually extracts features and then uses a translation layer that sort of mimics the summarizing aspect. These features are based on the buffer word and its surroundings as well as on the two previous annotated entities. These annotated words do not have to be in the surroundings of the buffer word though, they can be arbitrarily far back in the document, an aspect that ameliorates the behavior of a CRF model which is bound in the number of previous decisions that the model reads.

d) Prediction Layer

The overall framework of structured prediction is transition-based. This changes the perspective of having each word as the object of interest to be tagged, and introduces the concept of predicting transition sequences. Starting with an empty stack and all words in buffer, one action is chosen that will modify what is in stack and what is in buffer, as well as if there is any tag associated to it. With that, the goal is to predict the sequence of these actions. An example is shown in table 3.1 to illustrate the algorithm.

Table 3.1: Transition sequence example from Lamplet et al. [17]

Transition	**Output**	**Stack**	**Buffer**	**Segment**
	[ ]	[ ]	[Mark, Watney, visited, Mars]	
SHIFT	[ ]	[Mark]	[Watney, visited, Mars]	
SHIFT	[ ]	[Mark, Watney]	[visited, Mars]	
REDUCE(PER)	[(Mark Watney)-PER]	[ ]	[visited, Mars]	(Mark Watney)-PER
OUT	[(Mark Watney)-PER, visited]	[ ]	[Mars]	
SHIFT	[(Mark Watney)-PER, visited]	[Mars]	[ ]	
REDUCE(LOC)	[(Mark Watney)-PER, visited, (Mars)-LOC)]	[ ]	[ ]	(Mars)-LOC)

Adapted to spaCy's approach, they define a set of actions: shift, reduce, out and tagging (following a BILUO tagging scheme), to choose from in every transition. BILUO stands for "Beginning, Inside and Last (tokens of multi-token chunks), Unit-length chunks and Outside" so that there is differentiation between compound names and two separated names in a row for instance.

Translating these concepts into a prediction layer, a multi-layer perceptron is used to get the different action probabilities from the features obtained with the attention layer. Then, a validation check is done to discard those actions that are not suitable for the state and the best valid action is the one performed.

Given this whole architecture for spaCy's NER, there is also the option of using pre-trained language models. The concept of language models refers to contextualized word representa-

tions. Therefore, when we initialize the training of the model, the weights of the trigram-CNN layers do no start with random weights, but with the pre-trained ones. Regarding this, there are three different sizes of pre-trained CNNs of spaCy's NER for each language together with a blank option:

a) Blank: as its name indicates, the initialization of weights is completely random in all layers and the model begins from scratch. This is common to any language.

b) English:

- small (en_core_web_sm): trained on OntoNotes [spaCy's F-score = 85,43].
- medium (en_core_web_sm): trained on OntoNotes with GloVe vectors trained on Common Crawl [spaCy's F-score = 86,20].
- large (en_core_web_sm): trained on OntoNotes with GloVe vectors trained on Common Crawl [spaCy's F-score = 86,40].

c) French:

- small (fr_core_web_sm): trained on WikiNER and UD French Sequoia [spaCy's F-score = 83,42].
- medium (fr_core_web_sm): trained on WikiNER and UD French Sequoia with FastText vectors trained on Common Crawl [spaCy's F-score = 84,67].
- large (fr_core_web_sm): trained on WikiNER and UD French Sequoia with FastText vectors trained on Common Crawl [spaCy's F-score = 85,63].

d) Spanish:

- small (es_core_web_sm): trained on WikiNER and UD Spanish AnCora [spaCy's F-score = 89,41].
- medium (es_core_web_sm): trained on WikiNER and UD Spanish AnCora with FastText vectors trained on Common Crawl [spaCy's F-score = 89,84].
- large (es_core_web_sm): trained on WikiNER and UD Spanish AnCora with FastText vectors trained on Common Crawl [spaCy's F-score = 90,32].

In-House

Since spaCy models are not easily configurable in all details of their architecture, an in-house model architecture is built. This in-house model's architecture adopts to some extent the architecture of spaCy NER models, while giving complete comprehensive access for fine-tuning options.

While the embedding step remains as it is in spaCy, some minor changes are included for the language model training (residual CNN layers). The kernel size (n-gram input for residual CNNs) can be adjusted for each of the layers and the depth of these layers can be modified as well. During the experiments, a complete overview on the kernel variation is given.

The prediction layer is a dense layer with a categorical focal loss to compensate for the imbalance between classes (PHI vs no-PHI)[9]. Adhering to the latest trends for training deep neural networks for speeding up the process, Adam [16] is set as the learning rate method for our in-house predictive model.

Evaluation

To perform an appropriate fine-tuning and model assessment, a set of experiments is defined. Different parameters are evaluated on performance, both in terms of results and efficiency. Usually, evaluations are based on the metrics of precision, recall and F-score. Low precision would imply de-identifying terms should not be (such as medical terms) and low recall would translate into sensitive PHI not being removed. F-score is a measure of a test's accuracy. It is calculated from the harmonic mean of precision and recall, where the precision is the number of correctly identified positive results divided by the number of all positive results, including those not identified correctly, and the recall is the number of correctly identified positive results divided by the number of all samples that should have been identified as positive:

$$F_{score} = 2 * \frac{(precision * recall)}{(precision + recall)}$$

In the case of efficiency, training and test duration as well as model size is considered. Time is measured with the magic cell command of Jupyter (%%time) which displays the processing time together with wall time per executed cell. Models are saved as pickle files and their size is taken into account. These are very relevant criteria for our use-case since we are looking for efficient results, that are light and can be production-ready quickly.

In terms of experiments to be performed with the models, the following variables are either assessed separately (to isolate their effect) or jointly, in some cases, for better comparison (some of the tests only apply to one of the architectures):

a) Language model (spaCy): with three pre-trained language models per language and a blank model (with no previous training), spaCy presents 4 different ways of initializing the training of their model. Balance between performance and size of the models are taken into account.

b) Kernel size (in-house): at the language model training stage, n-grams are utilized as input to the residual CNNs (fixed trigrams for spaCy's model). We will then assess if taken different n-grams instead affects the performance of the entire in-house model.

c) Language (both): to assure that the model is scalable, we look for a consistent performance between all languages.

d) Gold vs Silver (spaCy): in order to address the necessity of performing the labor-intensive task of manually correcting annotations, a comparison between the gold and silver standard is shown.

e) Letter Case (spaCy): realism is another key component of our synthetic EHR generation pipeline and letter case is the main contributor. Hence, we will evaluate the difference in performance between different scenarios of Title Case, UPPERCASE and lowercase.

f) Data Augmentation (spaCy): with a limited amount of resources available, it is crucial to determine the amelioration of performance when augmenting the training dataset.

g) Dropout (spaCy): aiming for a maximum level of generalization, dropout determination will help on avoiding overfitting for the detected names of the model.

h) Architecture (both): a head-to-head comparison betweeen spaCy's and in-house NER. Differences in performance as well as on model sizes are shown, to determine if the in-house NER is performing as good as industry standards and then include it in the *de-PHIEHR* pipeline.

For the production of a new tool, it is not only important to consider the improvement of performance, but also to evaluate the economic aspect. To achieve this, early-stopping methods exist that prevent a model from continuing with training based on a specific threshold. For our use case, we are using the "patience" criterion, which stops the training if the validation loss has increased for a certain amount of iterations. In such cases, the model is over-fitting and validation loss has passed its minimum. To determine this "patience" threshold in our experiments, we perform a set of sample trainings and evaluate the behavior of the loss regarding local minima.

Chapter 4

Results & Discussion

4.1 EHRs Intake

To prepare for the translation of synthetic EHRs, Spanish records must be extracted from the row-oriented DBMS. With reference to the configuration file required for the random extraction, the following settings are introduced for our experiments:

- Filters: all are set to null since this project does not require for any specific terms or patient demographics, it is better to consider any kind of EHR coming from any service in the hospital.
- Amount of EHRs: according to previous tools developed by Savana Médica and considering the limitation of resources for the translation (1 hour = 750 records), 10.000 EHRs are enough for this project.
- Amount of characters: after manual exploration of a sample of EHRs, some errors are detected among the synthetic records. These faulty EHRs come with an error message in their raw text section. The message is about 2.500 characters long so the filter is set to 3.000 characters as minimum amount for a record.
- Filtered/Non-Filtered: the split is set to 0 since there are no filters used to get record IDs from the column-oriented DBMS.

By running the Python class with this configuration file, a set of 10.000 raw text records is stored as a list and saved as a pickle file ready for the translation. The entire process takes, in average, 46.8 seconds per 1.000 records extracted from the row-oriented DBMS, which in total adds up to almost 8 minutes to complete the whole set of 10.000 EHRs. With negligible times for extraction and an easily configurable settings file, this module of the pipeline proves to be easily scalable if there is need for data augmentation.

4.2 EHRs Translation

4.2.1 Pre-Processing

With the file of 10.000 EHRs ready, records are put through the pre-processing pipeline so that data cleansing together with data enrichment are applied to the dataset.

Language Detection

As previously explained, during the synthetic records creation in Spanish, due to the origin of the sources, noise is introduced to the system and some EHRs end up containing terms in Catalan. Therefore, the prepared module of *langdetect* is run and, after 5 different runs (due to its non-deterministic behavior), we take the seed with the maximum amount of identified as non-Spanish records. In this case, there are 98 records detected as Catalan EHRs from the whole set. These are pruned, and we end up with a file containing 9.902 records in Spanish.

Silver Standard

Getting ready for the translation, it is needed to pre-annotate the records so that we do not lose relevant information in the process. As a shortcut for the detection of names, the obsolete NER model in Spanish for name-detection is utilized.

Afterwards, the module for identifying street-related terms in the surroundings of detected names is executed. Among all annotations, only a few are related to locations. This is coherent with the high precision of the previous Spanish name-detection model. Indicated names are removed from the annotations in the dictionary of annotations, and a list containing the terms detected is saved for future analysis.

Next, the remaining tagged names are checked against the ENDs list. A process that runs almost instantly by coming back to the lookup table built. There happens to be no eponymous disease among the detected names, so all annotations are maintained after this step of pre-processing.

Avoiding the loss in translation, *Faustino* is used as a persistent term for all identified names. This pre-annotation process takes about 3 hours and 30 minutes in total: a rate of 47 records annotated per minute (1,25 seconds per record). After that, all 9.902 records are stored as raw text with no annotations associated. Throughout this process, a dataframe containing an overview of each record is generated. From this, the summary statistics shown in figure 4.1 are obtained.

```
syn_ehr_10k_v1
---------------
avg n. of annots:  5.3 ± 59.4
avg text length:   7073.1 ± 13923.7
avg pos of annots: 2053.2 ± 7692.1
-pctg of annots in q1:    32.26 %
-pctg of annots in q2-q3: 35.44 %
-pctg of annots in q4:    32.3 %
```

Figure 4.1: Summary statistics for 10.000 extracted EHRs

Since the idea of this experiment is to replicate a future stream of real records, we will maintain everything as mainstream as possible. To do so, a filtering on the records is performed with respect to their statistics. Summary information reveals that there is no real difference (in average) for encountering names in certain sections of an EHR, so position of annotations is not a determining variable for us. Regarding to the rest of variables, only records within the standard deviation for the text length are taken.

Moreover, number of annotations seems to fluctuate significantly through the dataset. Then, with a manual inspection of the records containing high amount of names (over 70 in some cases), it is detected that these tend to contain almost no medical information and the system has introduced a set of senseless sentences with only doctors and patients names recursively. Thus, only texts containing up to 10 detected names are taken for the experiment.

By filtering the 9.902 records with these parameters, we end up with 2.257 relevant records ready for translation. They get stored as a pickle file and lately will be loaded in the machine translation module.

4.2.2 Machine Translation

For this step, records are loaded and assigned to a variable as a list. This allows for the SDL translation server to properly queue the input for the four processors to work full time. As stated earlier, there are limited amounts of resources for the translation so that it is a race against the clock and only certain amount of records will be translated.

With the time limit set to three hours for each stage of the translation, the virtual machine used is manually monitored during the entire process. Once it is launched, it takes about 10 minutes for it to start translating documents. At a rhythm of about 750 records translated per hour, for the translation Spanish-to-English, 2.111 EHRs are processed by the translator. There are only 146 records that remain without translation due to the charging options of the translation service (per hour of activity).

Due to the structure of the translation pipeline, there are 2.111 EHRs available for English-to-French translation. In this occasion, the service is performed similar to the Spanish-to-English process, and all records got translated within the 3 hours. We end up then with 2.111 files containing Spanish, English and French raw texts.

4.2.3 Post-Processing

To get a useful training set, records are expected to be entirely annotated with all names detected. Not only that, but these synthetic EHRs must replicate real records as much as possible. For that end, two post-processing tasks are needed in the pipeline.

Gold Standard

After loading the 2.111 Spanish EHRs as XMI files in the annotation tool with all *Faustinos* annotated, a meticulous manual inspection is performed. Taking in average 45 seconds per record to get annotated, it adds up to more than 26 working-hours for the annotators to go through every single record.

Synthetic Random Naming

Considering that we already own a perfectly annotated set of raw texts in all three languages, for the model to fully learn based on context and not just memorizing usual personal names in a text, we substitute *Faustinos* by random names in each language, aiming for a zero-shot testing scenario. Another critical aspect is the realism of the EHRs, for this, the datasets of names and surnames in the three languages are utilized. Furthermore, a realistic frequency of title case, uppercase and lowercase naming is put in place.

With 2.111 records containing in average 6 annotated names (between names and surnames), over 10.000 terms are needed in each language. To lower loading times and overall project's size, the minimum safe amount of names and surnames are kept from each language database. This translates in 13.000 names and surnames for Spanish, English and French.

Regarding the letter case matter, the white-list generated from the gold standard generation is taken as a sample for computing frequencies of names casing. The frequencies obtained are the following:

- Title Case: 65%
- Lowercase: 2%
- Uppercase: 33%

This distribution is used as baseline, but different configurations are tested so that model behavior regarding letter case sensitivity is analyzed. Even though this brings realism to the records, it can be ameliorated if we consider that humans tend to follow same letter case rules within the same document, this will be presented as future work for synthetic record adaptation to new languages.

4.3 Hybrid System

4.3.1 Rule-based

To assess the performance of the PHI detection with the regex module defined, only unit tests can be taken as indicator. Keeping in mind that these tests are set by hand, a thorough research on all possible variations of each PHI type has been done. Initially, the matcher was returning additional characters with the designed token in the cases of having periods or commas before or after. By setting the boundaries of each rule appropriately, all tests are successfully met, and rules do not have to be adjusted until real EHRs for each language are tested.

```
{ 'info@helloworld.fr':  'info@helloworld.fr',
  'inf_o@helloworld.fr': 'inf_o@helloworld.fr',
  'info@helloworld.com': 'info@helloworld.com',
  'info@hello.world.fr': 'info@hello.world.fr',
  'InFo@HelLoWorlD.FR':  'InFo@HelLoWorlD.FR',
  ' info@helloworld.fr': 'info@helloworld.fr',
  '.info@helloworld.fr': 'info@helloworld.fr'
  'info@helloworld.fr ': 'info@helloworld.fr'
  'info@helloworld.fr.': 'info@helloworld.fr'
}
```

Figure 4.2: Rule-based unit testing for French emails

In figure 4.2, there is an example of the prepared unit testings for French emails detection. As we can see, all types of variations are considered as isolated cases so that misfits can be addressed individually. This proves to be an efficient way of setting up the regex for the rule-based module of the *De-PHIEHR*.

4.3.2 Deep Learning

For assessing the suitability of the different possible NER models prepared, most relevant variables are taken into account separately (to isolate their effect) and jointly in some cases for better comparison. All models share the same number of iterations for training: 20, with patience set to 5. In terms of train/test setup, a threefold cross-validation is done together with a 50/30/20 (train/test/validation) split.

Language Model (spaCy):

SpaCy provides with a set of pre-trained language models for each language in our research. There are three different options in terms of sizes (small, medium and large) and they differ in the depth of training. We address here if, for EHRs de-identification, it is worth implementing

the large pre-trained model at the cost of its much larger size (see table 4.1). All three languages come with similar sizes of pre-trained language models.

Table 4.1: Size (in MB) and Performance of spaCy's pre-trained language models in French (with standard configuration and 1.000 records)

Language Model	Size	Precision	Recall	F-score
spaCy_blank_fr	6.8MB	**92,86**	85,19	88,87
spaCy_small_fr	16MB	92,32	86,04	89,07
spaCy_medium_fr	54MB	91,77	90,25	91,01
spaCy_large_fr	604MB	91,83	**92,72**	**92,27**

In terms of performance, we see in table 4.1 that F-score presents a clear amelioration as we increase the size of the pre-trained language model. While it is below 90 F-score for blank and small, the model reaches more than 92 of F-score for the large one with just 1.000 EHRs for training. Additionally, we see that the contribution to this acceptable level of F-score comes from the equal contribution of both precision and recall.

Kernel Size (In-House):

Another minor variable that is present in both proposed architectures is the kernel size. It defines the amount of words to consider for the embedding at the input of each layer. While spaCy does not provide a straightforward method for tuning the parameter, in-house takes kernel size as a configuration variable for training and is the one tested for the discussion. To discuss the effect of it, two constant and one variable sizes are tested with the same configuration (see table 4.2).

Table 4.2: Performance of in-house model (kernel size) in French

Model Configuration	Kernel	Precision	Recall	F-score
inhouse_2111_fr	2	85,59	82,92	84,24
inhouse_2111_fr	3	85,64	**86,08**	85,86
inhouse_2111_fr	[2, 3]	**90,36**	83,07	**86,56**

While the model shows a slight improvement when trained with variable kernel size, we see that this leads to a significant unbalance between precision and recall, an issue very problematic for Savana Médica's use case. Keeping this in mind and looking at the steady performance of a kenel size of 3 (same as spaCy), we go for this alternative.

Language:

Focusing on the main concern of the research, the model must be functional and reach acceptable performance in any language. For that, the same architecture is tested for French, English and Spanish in both spaCy's and in-house models.

a) SpaCy:

From figure 4.3, we can appreciate that all three languages undergo a very similar training process, reaching the epoch number 20 at a similar level of losses. Regarding to the time taken, they all took almost identical wall time, showing that language is not a limitation when it comes to training.

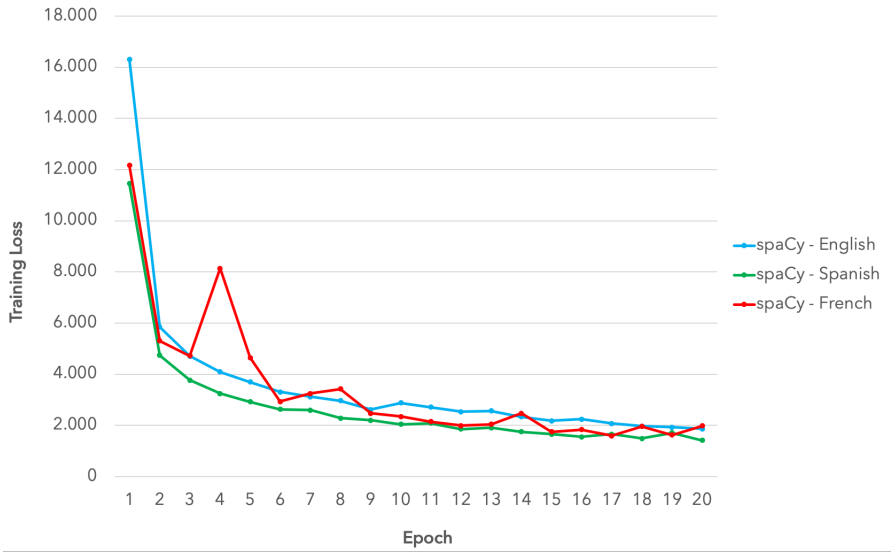

Figure 4.3: Training losses of spaCy's model (language)

In table 4.3, we can have a glance at the F-score levels for each language when using the exact same configuration of the model.

Table 4.3: Performance of spaCy's model (language)

Model Configuration	Language	Precision	Recall	F-score
spaCy_blank_2111	English	**93,15**	87,89	90,44
spaCy_blank_2111	French	91,86	90,53	91,19
spaCy_blank_2111	Spanish	91,11	**92,12**	**91,61**

b) In-House:

Even though the behavior is expected to be similar for both models in terms of multilinguality, the same experiment is run for the in-house model (see figure 4.4). Training shows complete alignment between three languages, although losses are not comparable with spaCy due to the loss function put in place for each model.

With reference to the overall F-score of each model (see table 4.4), we again see that they are fairly comparable and the model can be equally used for any of them.

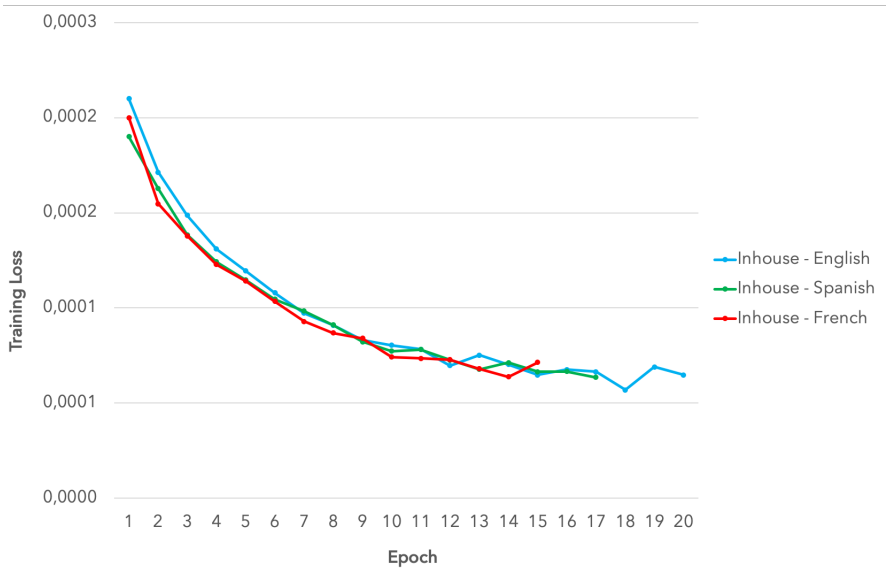

Figure 4.4: Training losses of in-house model (language)

Table 4.4: Performance of in-house model (language)

Model Configuration	Language	Precision	Recall	F-score
inhouse_2111	English	**86,45**	85,55	85,99
inhouse_2111	French	85,64	**86,08**	85,86
inhouse_2111	Spanish	86,21	85,88	**86,04**

Gold vs Silver:

In order to assess the necessity of performing the labor-intensive task of manually annotating, a comparison between the gold and silver standard are shown. From table 4.5, we see an improvement in both precision and recall of 2 percentage points, making clear that "goldenization" of the records is worth the time taken.

Table 4.5: Performance of spaCy's model (gold vs silver)

Model Configuration	Standard	Precision	Recall	F-score
spaCy_blank_1000	Silver	90,06	83,02	86,39
spaCy_blank_1000	Gold	**92,50**	**85,32**	**88,77**

Letter Case:

Realism is another key component of our synthetic EHR generation pipeline, and letter case is the main contributor. Hence, we will evaluate the difference in performance between two different scenarios: only title case vs realistic letter case (with computed frequencies).

Table 4.6: Performance of spaCy's model (letter case)

Model Configuration	Letter Case	Precision	Recall	F-score
spaCy_small_fr_1000_0,5	Title	83,62	**91,69**	87,47
spaCy_small_fr_1000_0,5	lower/UPPER/Title	**88,34**	88,81	**88,58**

Keeping the realistic frequencies for the variable setup, we obtain certainly different behaviors between both (see table 4.6). Title case, considering that a key feature for the embedding is the shape of the word, considerably ameliorates the recall. However, the model is clearly overshooting, since the levels for precision are much lower due to the model identifying as name every word that is in title case. This brings up the necessity for generating the realistic frequencies for the training set so that the model is better prepared for future de-identification of real EHRs.

Data Augmentation:

With a limited amount of resources available, it is crucial to determine the variation in performance when augmenting the training dataset. As expected (see table 4.7), F-score experiences a clear improvement with data augmentation, we cannot anticipate when it will reach a plateau so that more records are needed and results will likely keep improving. Nevertheless, a bigger training set comes with a cost; we see a linear increase in the training times.

Table 4.7: Performance and Training Time of spaCy's model (data augmentation)

Model Configuration	No. Records	Precision	Recall	F-score	Training Time
spaCy_large_en	1000	92,86	85,19	88,86	58min 36s
spaCy_large_en	1500	92,75	87,23	89,91	1h 21min 17s
spaCy_large_en	2111	**93,15**	**87,89**	**90,44**	1h 57min 45s

Dropout:

Aiming for a maximum level of generalization, dropout determination will help on avoiding overfitting for the detected names of the model. Testing out the parameter at 0,2 and 0,5 on the predictive layer, we can see in figure 4.5 and table 4.8 that with lower dropout the model ameliorates the training performance and still brings better testing performance. Therefore, a dropout of 0,2 is proposed for the final model configuration.

Table 4.8: Performance of spaCy's model (data augmentation)

Model Configuration	Dropout	Precision	Recall	F-score
spaCy_blank_fr	0,2	91,11	**92,12**	**91,61**
spaCy_blank_fr	0,5	**92,50**	85,32	88,76

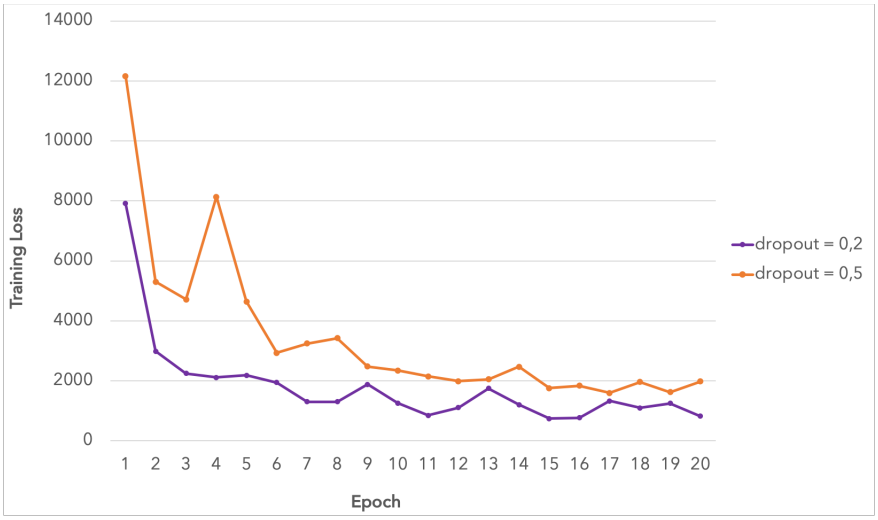

Figure 4.5: Training losses of spaCy's model (dropout)

Architecture:

Lastly, an overview on the performance of both models is put together (see table 4.9). While spaCy counts with a set of options to choose from, it is only fair to compare performances between the blank model of spaCy and in-house model. Reason behind this is that they both initialize with random weights in all layers, not having a pre-trained language model.

Even though the attention layer is missing in the in-house solution, results are fairly comparable and we can state that our architecture, with certain additions, can compete with spaCy's NER. Model weight is still far from being similar but this is caused by many factors that will be refined with time.

Table 4.9: Size (in MB) and Performance of both architectures in French

Language Model	Size	Precision	Recall	F-score
spaCy_blank_fr	6.8MB	**92,86**	**85,19**	**88,87**
in-house_fr	166MB	90,36	83,07	86,56

All in all, we have encountered a configuration for the in-house model that can compete with state-of-the-art approaches for industrial use cases (spaCy). With a limited amount of EHRs, 2.111, the in-house NER is detecting most names present in the text, most likely missing rare cases, an aspect that can be ameliorated by augmenting the amount of training data. Not only this, but the model is performing at such level with a zero-shot learning methodology in place. This means that the model is purely learning from context and shape of the name, not profiting from the norm, prefix and suffix of the embedding.

Chapter 5

Conclusions

5.1 Accomplishments & Limitations

The purpose of this study was to test whether we can build a state-of-the-art de-identification software for different target languages using synthetic and limited data. Below, we conclude the important findings and implications for future work.

Production-Ready

We have developed a pipeline that can easily be run in a production environment. For that to be possible, we encapsulated the entire code as a set of classes with dedicated methods. In addition, we integrated configuration files in a comprehensive format so that adding of functionality or adjustments can be made ad-hoc.

Important features of any software are error and performance tracking. In the de-PHIER, the modularity of the implemented code as well as unit tests, allow for easy debugging and control of proper functionality for both the NER model and the regex module.

Even though we did not manage to cover all PHI classes in this work, the design of the de-PHIER enables to integrate additional classes seamlessly. Obviously, complex PHI classes such as names do require the preparation of a different training set and re-training of the NER model to add those classes. The rule-based system on the other hand, can be updated by just adding new patterns.

Our NER model detects names with an F-score higher than 85%, however, this value is still below the acceptance criterion to enter a production state. Nevertheless, this work renders the tested approach as useful and reaching a higher F-score is only a matter of data augmentation and testing also other NER architectures. A positive aspect of our in-model is that it is very memory-efficient and can be easily integrated in any workflow just as spaCy's models, which are among the most efficient for industrial applications.

Multilingual

Savana Médica's expansion to other countries comes with the need to adapt NLP software to different languages. Therefore, among many other tools, a universal PHI detection framework valid for any language is also needed. As of today, no models exist that can deal with multiple languages at once and at the same time be easily manageable by maintaining the required quality. Although recent developments have come up with embeddings that aim for the unification of multiple languages in one vector space [4], such embeddings are usually trained on public resources and not useful for the clinical domain.

Thus, training language specific NER models is still state-of-the-art and by using a back-end design, as the one presented here, the orchestration of multiple language specific models in a single pipeline still seems the better solution. By doing so, new languages can be added to the de-PHIER in a reasonable amount of time and, in case a specific model needs to be adjusted, this can be achieved in a more controlled setting.

In this work, we showed that our approach for a multilingual solution is feasible by applying this design on the three languages English, French and Spanish for testing. Results revealed similar performance in terms of F-score and training time for all three languages, thereby assuring that the overall architecture is language-agnostic.

Low-Resources

Many potential clients in other countries request a de-identification solution before engaging with Savana Médica. This means that before actually having access to any EHR in the new language, the tools need to exists to de-identify EHRs in that very language. Thus, the de-identification tool, and especially the NER models here, need to be developed without real data. Savana Médica solved this issue for Spanish records by generating a set of synthetic EHRs manually with a pool of medical doctors. Nonetheless, this workflow is costly and time-consuming, and thus not an efficient solution for new languages.

Even so, by taking advantage of this existing database of synthetic Spanish records, we applied a translation pipeline and showed that this approach is efficient and results in high quality results. While there are some open-source libraries that can perform this machine translation task, we used the cloud computing space of AWS using the SDL translation service which has proven to be a reliable translation tool. For this pilot study we translated 2.111 EHRs, which was sufficient to prove that our approach is working. We were able to train NER models obtaining results above 90% of F-score. To upscale this approach a bigger amount of documents needs to be translated and more money invested in translation.

5.2 Future Work

This project was a proof on concept (PoC) to create a full pipeline for multilingual de-identification. However, there is room to improve our hybrid system. Among the most important aspects to consider for future work are the ones outlined below:

- Data Augmentation: to improve the in-house NER performance, a richer and larger EHR corpus is needed. Thus, an additional set of records should be extracted and translated.
- Multilingual Word Embeddings (MWE): recently, new approaches appear that deal with multilingual setups. Even though managing specific language models has advantages over those, such approaches might open new possibilities for problems such as de-identification in many different languages. Models such as the multilingual-BERT[1] or MUSE[2] are exploring the boundaries of what is possible today.
- BILUO: in this project, we used a binary tagging scheme of name (PER) or no-name (O) for the in-house NER. Nevertheless, with a transition sequence predictive layer, we lose predictive power because we do not differentiate between names and surnames. This can be implemented with a BILUO tagging scheme incorporation in future versions.
- Partial F-score: some of the manually curated names did not match the tokenization schema applied. Due to those inconsistencies between curated and predicted names, the F-score was a bit lower than it could have been. Thus, care has to be taken that curations and predictions are aligned. This can be achieved by providing better guidelines for manual curation, adapt the tokenization approach, or by relaxing the calculation of the F-score by treating overlapping annotations of the same PHI class between prediction and true value as an agreement.

Our approach is a first step towards solving some of the issues related with adapting NLP tools, in this study a de-identification tool, to different languages. We aimed at a combination of regexes and a NER model and we developed both based on synthetic data translated to different languages. By combining regexes and the NER model in a flexible back-end architecture, multiple language-specific regex modules and NER models can be integrated into one tool which is easily configurable and applicable to EHRs in different languages.

[1] https://github.com/google-research/bert/blob/master/multilingual.md
[2] https://github.com/facebookresearch/MUSE

Bibliography

[1] Absolutdata. Experience Extended — Fuzzy String Matching, 2020. 7

[2] Duy Duc An Bui and Qing Zeng-Treitler. Learning regular expressions for clinical text classification. *Journal of the American Medical Informatics Association*, 21(5):850–857, 2014. 8

[3] CDC. Health Insurance Portability and Accountability Act of 1996 (HIPAA), 2018. 3

[4] Xilun Chen and Claire Cardie. Unsupervised Multilingual Word Embeddings. 8 2018. 37

[5] Aravind CR. Word Embeddings in NLP, 2020. 9

[6] Franck Dernoncourt, Ji Young Lee, Ozlem Uzuner, and Peter Szolovits. De-identification of Patient Notes with Recurrent Neural Networks. 6 2016. 9, 10

[7] FDA. Real-World Evidence, 2020. 2

[8] Oscar Ferrández, Brett R South, Shuying Shen, F Jeffrey Friedlin, Matthew H Samore, and Stéphane M Meystre. Evaluating current automatic de-identification methods with Veteran's health administration clinical documents. Technical report, 2012. 8

[9] Umberto Griffo. Focal Loss, 2020. 24

[10] HealthIT. What is an electronic health record (EHR)?, 2019. 1

[11] Ignacio Hernandez Medrano, Jorge Tello Guijarro, Cristobal Belda, Alberto Urena, Ignacio Salcedo, Luis Espinosa-Anke, and Horacio Saggion. Savana: Re-using Electronic Health Records with Artificial Intelligence. *International Journal of Interactive Multimedia and Artificial Intelligence*, 4(7):8, 2018. 2

[12] HL7 International. Introduction to HL7 Standards, 2016. 2

[13] Investopedia. General Data Protection Regulation (GDPR), 2020. 3

[14] Investopedia. Luhn Algorithm, 2020. 20

[15] Kaung Khin, Philipp Burckhardt, and Rema Padman. A Deep Learning Architecture for De-identification of Patient Notes: Implementation and Evaluation. 10 2018. 10

[16] Diederik P Kingma and Jimmy Lei Ba. ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION. Technical report. 24

[17] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural Architectures for Named Entity Recognition. Technical report. xi, 22

[18] Scikit Learn. Stochastic Gradient Descent, 2020. 9

[19] Zengjian Liu, Buzhou Tang, Xiaolong Wang, and Qingcai Chen. De-identification of clinical notes via recurrent neural network and conditional random field. *Journal of Biomedical Informatics*, 75:S34–S42, 11 2017. 11

[20] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in Translation: Contextualized Word Vectors. 7 2017. 10

[21] Foresee Medical. What Is EHR Software? And Must-Have Features for Interoperability, 2020. 1

[22] Stephane M. Meystre, F. Jeffrey Friedlin, Brett R. South, Shuying Shen, and Matthew H. Samore. Automatic de-identification of textual documents in the electronic health record: A review of recent research, 2010. 8

[23] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. Technical report. 9

[24] MonkeyLearn. Named Entity Recognition: Concept, Guide and Tools, 2020. 7

[25] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. 2 2018. 10

[26] SAS. Natural Language Processing, What is and why it matters, 2020. 2

[27] spaCy. Models - Model Architecture, 2020. 10

[28] Amber Stubbs, Christopher Kotfila, and Özlem Uzuner. Automated systems for the de-identification of longitudinal clinical narratives: Overview of 2014 i2b2/UTHealth shared task Track 1. *Journal of Biomedical Informatics*, 58:S11–S19, 12 2015. 7

[29] Expert System. What is Machine Learning? A definition, 2020. 8

[30] Özlem Uzuner, Yuan Luo, and Peter Szolovits. Evaluating the State-of-the-Art in Automatic De-identification. *Journal of the American Medical Informatics Association*, 14(5):550–563, 9 2007. 7

[31] Hui Yang and Jonathan M. Garibaldi. Automatic detection of protected health information from clinic narratives. *Journal of Biomedical Informatics*, 58:S30–S38, 12 2015. 11