Information extraction with mBERT from a self annotated dataset

Marc Ros Martí

ieConsumo S.L.

Supervisors:
Vicenta Bosch Ayala
Lluís Padró Cirera

A thesis submitted for the degree of

Master in Artificial Intelligence

Barcelona, January 2022

This thesis is submitted to the Facultat d'Informàtica de Barcelona, Universitat Politècnica de Catalunya in fulfilment of the requirements for the Master in Artificial Intelligence.

Marc Ros Martí, January 2022

Acknowledgements

I would like to thank Tica and Josep for trusting me to develop this project. I'm grateful to have learned from you and to have contributed to your long-term work in the field. I would also like to thank Lluís Padró for advising me on this project. Finally, thanks to my family for supporting me all this time and, especially, to Laura, for being my best working (and non-working) partner.

Abstract

In this project, we present an approach to automatically extract relevant information (Named Entity Recognition) from clinical courses. One of the main particularities of this work is that we performed the whole task end-to-end: from collecting and annotating the data to training a deep learning model. After determining the most relevant labels for the task, we generated the dataset by annotating the raw text of the clinical courses using an in-house annotation tool. The resulting dataset is composed of 3501 annotated sentences in Spanish. Although this amount of data should be enough to perform the task, it suffers from class imbalance. We overcame this problem by applying Data Augmentation techniques for NLP. Using all this data, we fine-tuned a pre-trained distilled multilingual BERT (DistilmBERT) model. Our best-performing model uses augmentation techniques (combining both Back-Translation and Word Replacement with Contextualized Word Embeddings) and a Linear layer model head. It achieves a macro F1-score of 86.39% on the test set, 1.43pp better than our best model with the non-augmented dataset, which proves the worthiness of the augmentation solution.

Keywords: NLP, NERC, Annotation, Transformers, BERT, mBERT, DistilmBERT, Data Augmentation

Resum

En aquest treball presentem una proposta per extreure informació rellevant de manera automàtica (Named Entity Recognition) de cursos clínics. La gran particularitat d'aquest projecte és que hem realitzat la tasca sencera, d'inici a fi: des de la recol·lecció i anotació de la dataset fins a l'entrenament d'un model de Deep Learning. Després de determinar quines eren les etiquetes més rellevants per a la tasca, hem generat la base de dades tot annotant el text lliure dels cursos clínics fent ús d'una eina d'anotació d'elaboració pròpia. La base de dades consta de 3501 oracions anotades en castellà. Tot i que aquesta quantitat de dades podria ser suficient per a realitzar la tasca, la base de dades pateix de class imbalance. Per aquesta raó, vam decidir intentar solucionar el problema aplicant tècniques de Data Augmentation per a NLP. Fent ús de totes les dades finals, hem fet fine-tuning d'un model distilled multilingual BERT (DistilmBERT) pre-entrenat. El nostre millor model fa ús de data augmentation (combinant Back-Translation i Word Replacement amb Contextualized Word Embeddings) i una Linear layer com a model head. Aquest model aconsegueix un valor de 86.39% de macro F1-score a les dades de test, que és 1.43pp millor que la nostra millor proposta sense augmentar les dades. Aquests resultats demostren la utilitat d'aplicar tècniques d'augment de dades en el nostre projecte.

Paraules clau: NLP, NERC, Anotació, Transformers, BERT, mBERT, DistilmBERT, Data Augmentation

Resumen

En este trabajo presentamos una propuesta para la extracción automática de información relevante (Named Entity Recognition) de cursos clínicos. La gran particularidad de este proyecto es que hemos realizado la tarea de inicio a fin: des de la recogida y anotación de los datos hasta el entrenamiento de un modelo de Deep Learning. Después de determinar cuáles eran las etiquetas más importantes para nuestro trabajo, hemos generado la base de datos anotando los cursos clínicos (en texto libre) con una herramienta de anotación de elaboración propia. La base de datos final consta de 3501 oraciones anotadas, todas ellas en castellano. A pesar de que este número de datos podría parecer suficiente, la base de datos está claramente no balanceada. Por esta razón, hemos solucionado el problema aplicando técnicas de Data Augmentation para NLP. Haciendo uso de todos los datos finales, hemos hecho fine-tuning de un modelo distilled multilingual BERT (DistilmBERT) ya pre-entrenado. Nuestro mejor modelo hace uso de data augmentation (combinación de Back-Translation y Word Replacement con Contextualized Word Embeddings) y una Linear layer como model head. Este modelo consigue una puntuación macro F1 de 86.39% en la base de datos de test. Este resultado es 1.43pp mejor que nuestra mejor propuesta sin aumentar los datos. Por consiguiente, estos resultados demuestran la aplicabilidad de técnicas de aumento de datos en nuestro proyecto.

Palabras clave: NLP, NERC, Anotación, Transformers, BERT, mBERT, DistilmBERT, Data Augmentation

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Natural language processing (NLP) is a subfield of Artificial Intelligence (AI) that deals with the interaction between computers and human language. Its main goal is to program computers able to understand the contents of documents so as to perform a wide range of tasks [55]. Among them, information extraction (IE) is one of the most prominent. It consists in extracting (automatically) structured information from unstructured, semi-structured or structured machine-readable documents. Inside the field of IE, there are several specific tasks that can be performed like Named Entity Recognition (NER), relation extraction, event extraction or template filling. In this work, the task to be performed is NER, and more concretely Named Entity Classification (NEC/NERC). Prior to defining the goal of NERC, it is necessary to clarify what a named entity is. A named entity is a real-world object that can be referred to with a proper name [56]. Importantly, a real-world object is not solely restricted to physical objects but also to abstract. Thus, some of the more typical entities are both physical (persons, locations, organizations, products, etc.) and non-physical (dates, times, temporal expressions, etc.). These are only some generic examples of named entity types; depending on the use case, new application-dependent ones can be defined. Said that, NERC aims to find each mention of a named entity in a given text and classify it with its corresponding label [21].

The preferred algorithms to perform NERC have evolved over time from handcrafted rule-based systems to more complex statistical and machine learning approaches. The current state-of-the-art for the task is using Neural NERC, i.e. the text (usually tokenized) is used as the input of a neural network. Among the different Neural NERC approaches, the Transformer-based models [50] like BERT [7] are the current hot-topic in the field. These models are usually hard and expensive

to train from scratch, e.g. BERT, without being one of the biggest models at all, has 335M of parameters and requires massive amounts of data. A way to solve these limitations is applying transfer learning, usually taking a pre-trained model in a general-purpose corpus and fine-tuning it to the specific area of application. Bear in mind that Transformer-based models are not originally intended for downstream tasks (like NERC). Thus, its architecture needs to be adapted depending on the task. In the case of NERC, it is necessary to add either a Fully-Connected or CRF layer [47] on top of it, which allows performing classification tasks at token level.

IE is probably the most used NLP task in industry and, inside IE, NERC is the most typical application. Indeed, this report presents the application of NERC inside a company: ieConsumo S.L. [23]. ieConsumo S.L is an SME specialized in product safety management. Among their areas of expertise, they deal with product risk assessment. They offer their services to companies that need to recognize and identify possible hazards that their product may have and evaluate them.

Their current effort is to develop PSM [35], a tool that helps to extract meaningful product safety data (e.g from users' complaints) that can be used to improve products, reduce their risk and prevent possible recall in the future. There are several multinational corporations interested in acquiring their application. In the meanwhile, ieConsumo aims to make PSM even more competitive and appealing to potential clients. One of its main disadvantages is that currently the application is quite manual. Concretely, given a source of data, it requires an operator to detect and introduce one by one the necessary inputs for the application to work. Among these inputs there is the product involved, the type and severity of the injury, the circumstances around the incident, etc. In such a scenario, a good way to automatically detect these elements is to apply NERC. Indeed, this was the goal of the project: advance in the process of automatically populating PSM applying NLP.

The initial idea for this project was to use data from the complaints of potential customers. Like this was not possible, the project was developed using a similar field of application: clinical courses where a product is involved in an injury. For that, we took advantage of a project previously conducted by the company: PIPADI [22]. The objective of PIPADI is, in their own words "build an information collection system that allows us to analyze the conditions in which an accident took

place, study the possible risk factors associated with accidents and in the cases where an object was involved, proceed to analyze it". Several hospitals participated in the program. Most of the available clinical courses came from Hospital de Nens de Barcelona (pediatrics) [6] and Hospital de la Vall d'Hebron (burns) [10]. The available data was basically a set of free-text descriptions that needed to be conveniently annotated so as to perform the NERC task.

Given all the necessary context, we can present our methodology. First, according to the needs of the company, we defined the named entity labels that needed to be extracted from the text. After that, the data was annotated, generating our self-annotated dataset (PIPADI dataset). Then, to gain familiarity with the latest trends of NLP, we used this data to fine-tune a transformer-based model for the NERC task. Given that the available dataset was in Spanish, we had to select a multilingual transformer-based model, like multilingual BERT [8] (extension of BERT pre-trained in 104 languages instead of only English) or XLM-R [12]. These models are able to handle multiple languages, Spanish among them. See how using a multilingual model is not only convenient given the available dataset but also to add value to potential customers, especially if they are multinational companies that need to deal with complaints from users from all over the world.

1.1 Objectives

General objective: The overall aim of this Master Thesis is to automatize the process of extracting relevant information from clinical courses by means of implementing an NLP model for information extraction (NERC).

Specific objectives:

- To determine the most relevant entities to extract from the text.
- To find an annotator tool according to the company requirements.
- To annotate the dataset.
- To implement a solution fine-tuning a transformer-based multilingual model.
- To test and assess the solution with a test set.

Chapter 2

Literature Review

2.1 Annotation

In machine learning, data labeling is the process by which one or more labels are associated with raw data. It is an essential task as it generates the necessary information for a machine learning model to learn. Data labeling is a mandatory step for most of the fields of Artificial intelligence, from computer vision to NLP.

In NLP data labelling is also referred to as annotation. Depending on the task, the type of annotation is going to vary considerably, going from sentence-level labels to concrete token-level labels. For NERC, the objective task in this Master Thesis, the annotators will need to assign a label to each of the sentence tokens. For that, the BIO (or IOB2) format [37] is a common approach. Basically consists in adding *B* or *I* prefix before a named entity label. The *B*- prefix indicates the beginning of a named entity. The *I*- prefix indicates that the tag is inside a named entity. Finally, *O* indicates that a token does not belong to a labelled named entity [53]. This is the format employed but some of the most well known NERC datasets, like CoNLL-2002 [48] or CoNLL-2003 [49].

To make this process easier for annotators, there are several free labeling tools for text annotation in NLP. Some of the best are Doccano [11], Brat [31] or INCEpTION [9]. Doccano is probably the simplest of the three but it does not support relational annotations. If you need more functionalities, Brat looks like a convenient solution. Finally, INCEpTION is less straightforward than the prior tools; however, it supports PDF files and provides more features [19].

2.2 Transformers

Vaswani et al [50] presented Transformers in 2017. Transformers are sequence transduction models that completely get rid of recurrence and convolutions, as they are only based on attention mechanisms [1]. These models, by taking full advantage of the attention mechanism, can find global dependencies in the input or output sequence, regardless of their distance. In terms of efficiency, fully relying on attention allows much more parallelization than with recurrent models (which are inherently sequential algorithms). Concerning convolutional models, despite being more parallelizable than recurrent models, the number of operations to relate two arbitrary positions grow depending on the distance between them. This makes it more complex to learn dependencies between distant points. With Transformers this is not an issue, as in the attention mechanism the number of operations is reduced to a constant value.

As some of the most competitive neural sequence transduction models, Transformers follow an encoder-decoder structure. Note that transduction models are used in NLP basically for machine translation tasks. Indeed, the authors of [50] focused their experimentation only on this type of task. The encoder processes the tokens in the input sequence, generating a vector (context vector) that contains meaningful information about them. This vector is later fed into the decoder, which will generate an output sequence of symbols (one at a time). Note that these models are auto-regressive, i.e. the inputs for the decoder are not only the outputs of the encoder but also the symbols previously generated by the decoder itself.

Figure 2.1 shows the Transformer model architecture.

- Encoder. Left hand side of Figure 2.1. The proposed architecture includes a stack of 6 identical layers, each of them composed of two sub-layers: a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. Both modules are connected using residual connections. The output of each sub-layer can be defined as $LayerNorm(x + Sublayer(x))$, being $Sublayer(x)$ the function implemented by each sublayer.

- Decoder. Right hand side of Figure 2.1. Also composed of a stack of 6 identical layers. However, there are relevant differences regarding the multi-head attention layers. On the one hand, there is a third sublayer type that will

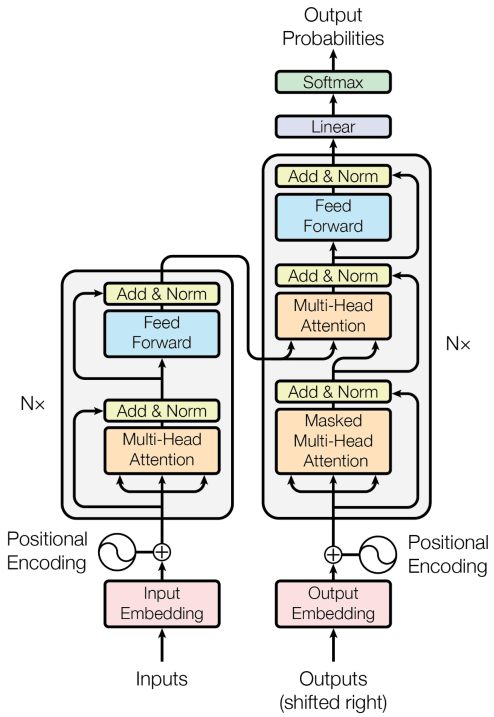

Figure 2.1: Transformer model architecture

be in charge of applying multi-head attention to the output of the encoder stack (context vector). On the other hand, see how in the decoder stack, right after the positional encoding, instead of applying a "Multi-Head Attention" module they used a "Masked Multi-Head Attention" module. This modified module ensures that the predictions for a concrete position do not have information of subsequent positions (preserving the auto-regressive nature of encoder-decoder architectures). As its name suggests, this module will simply mask all the positions after the current one, in order to prevent attention from taking future elements into account.

The following subsections analyse the operation of the sub-layers that compose the encoder and decoder stacks.

2.2.1 Attention Layers

Attention mechanism was first introduced in [1] to improve the performance of encoder-decoder models for machine translation. They used it to allow the decoder to weigh the input vectors according to their relevance, giving higher weights to the most relevant ones [2]. In Transformers, the idea is still the same: learn

(weigh) contextual relationships between elements in the sequence. In this case though, it is applied (with minor changes) to both the encoder and the decoder, as seen in the analysis of Figure 2.1.

When it comes to its implementation, every input needs to have three representations: key, query, and value [42]. An attention function maps these representations into an output. The output is the weighted sum of the values, being the weight per each of them the result of applying a compatibility function between the key and the query (the more compatible, the more weight, i.e the more relevant is a certain element in the sequence). Note that queries and key-value pairs are all vectors. Therefore, in order to improve the efficiency of the implementation, they can be packed into matrices (Q, K, and V, respectively) that allow to process them simultaneously, taking advantage of highly optimized matrix multiplication implementations.

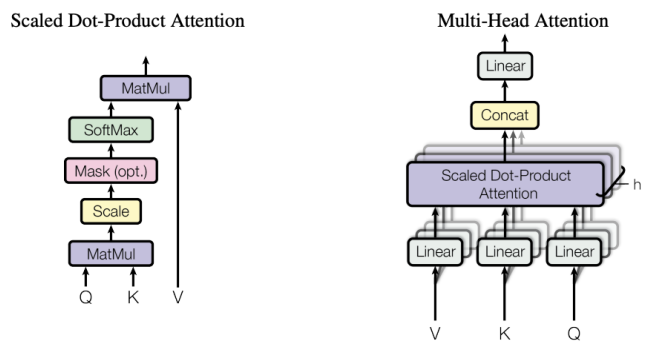

Figure 2.2: (left) Scaled Dot Product Attention. (right) Multi-Head Attention

As shown in Figure 2.2 the authors of [50] decided to use Multi-Head Attention in their implementation. For that, they presented Scaled Dot Product Attention, their implementation of the attention mechanism:

- Scale Dot Product Attention. Left hand side of Figure 2.2.

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

  The novelty of this definition with respect to the standard dot product attention is the scaling factor d_k (dimensionality of keys and queries). This is done to prevent the dot product from increasing a lot in magnitude. Otherwise, it

pushes the softmax to regions where it saturates, which makes the gradients very small (stops learning)

- Multi-Head Attention. Right hand side of Figure 2.2. The authors discovered that it was beneficial to perform the Scale Dot Product Attention several times instead of a single one. For that, they linearly project Q, K, and V matrices h times (where h is the number of multi-heads). Then, Scaled Dot Product attention is applied in parallel for all of them. Finally, the resulting versions are concatenated and projected to the original space. It is worth mentioning that to prevent this system from increasing the computational cost of the model, the dimensions of the representations are divided by the number of attention heads h. This way, the total computational cost is similar to performing a typical single-head attention.

2.2.2 Feed Forward Networks

As shown in Figure 2.1 both the encoder and the decoder include a fully connected feed-forward network right after the attention layer block. This network applies to each position of the sequence separately and identically:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2$$

As shown in the expression above, this network is nothing but two linear transformations with a ReLU activation function in between of them. In the case of the decoder block, after the last Feed Forward Network, the decoder output goes through a last Linear Transformation and Softmax function to convert the output into next-token probabilities (language modelling).

2.2.3 Positional Encoding

Unlike recurrent or convolutional models, Transformers do not have a mechanism to take advantage of the order of the sequence (the attention mechanism gets rid of it). To solve this issue, the authors introduced positional encodings. These encodings are present at the bottom of the encoder and decoder stacks and have the same dimension as the token embeddings, allowing all of them to be summed (see the bottom of the encoder and decoder stacks in Figure 2.1). The positional encodings

are generated through these formulas:

$$PE_{pos,2i} = sin(pos/1000^{2i/d_{model}})$$
$$PE_{pos,2i+1} = cos(pos/1000^{2i/d_{model}})$$

Where *pos* is the position and *i* is the dimension

2.3 BERT

Transformers have been a revolution in NLP. Since its introduction, several transformer-based models have been presented. Among them, BERT (Bidirectional Encoder Representations from Transformers) [7] is probably the one that has gained more notoriety. Note how, as its name indicates, it does not use the full Transformer architecture, but only the encoder block. BERT is designed to pre-train deep bidirectional representations from text in an unsupervised manner. In addition, applying transfer learning (fine-tuning), its pre-trained implementation can easily deal with a wide range of NLP tasks. It only requires the addition of a suitable output layer/s. In fact, applying transfer learning to pre-trained language models was nothing new. There were feature-based approaches like ELMo [34] that were using the pre-trained representations as input features to other architectures. And, like BERT, there were also fine-tuning approaches (e.g. GPT [36]) adapting to the specific task by fine-tuning all the pre-trained parameters (and adding some task-specific ones). However, the issue for both approaches was that their language representations were unidirectional (e.g. left-to-right in GPT). BERT overcomes this limitation thanks to Masked Language Modeling, their pre-training strategy (see 2.3.2).

Figure 2.3 shows how, after pre-training the model, it can be adapted to a wide range of downstream tasks by only introducing some minimal task-specific changes. For the NERC task (purpose of this project), it is enough to add a Fully Connected Layer or a CRF layer (with Softmax activation) on top of BERT's original architecture.

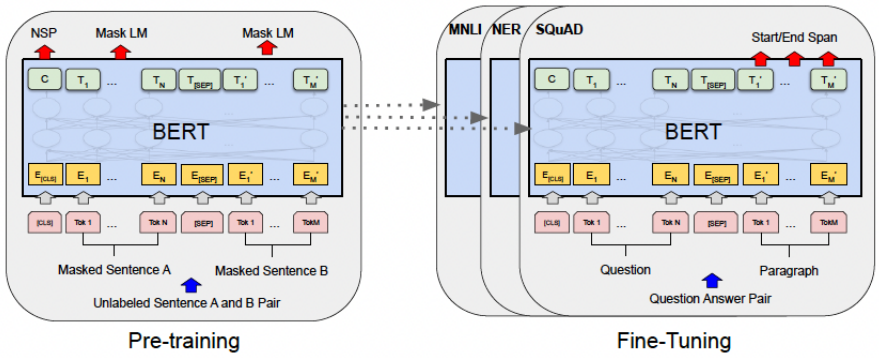

Figure 2.3: BERT framework: pre-training and fine-tuning

The authors proposed two different model sizes for BERT:

- $BERT_{base}$: $L = 12, H = 768, A = 12$. Replicates GPT. 110M parameters.
- $BERT_{large}$: $L = 24, H = 1024, A = 16$. 335M parameters.

Where L are the number of layers or Transformer Blocks, H is the hidden size and A is the number of attention heads.

2.3.1 Input Representation

In order to handle different types of tasks (regardless of being sentence or token level), the authors proposed an input representation that unambiguously represents token sequences for both single and pairs of sentences. Note how for the authors, *sentence* is defined as an arbitrary span of contiguous text, while *sequence* is the result of tokenizing the *sentence* (either a single sentence or two sentences packed together). BERT takes *sequences* as input.

For the tokenization, BERT uses WordPiece subword-based tokenizer [59] with a 30K token vocabulary. Subword-based tokenization is a solution between word (the most common) and character-based tokenization. It works by splitting rare words (not frequent in the training corpus) into smaller meaningful subwords, e.g. *boys* may be split into *boy* and *##s* (subword that denotes plurals) [41]. Apart from that, BERT includes a couple of special tokens: [CLS] and [SEP]. [CLS] is always the first token of every sequence. It is intended to be used for classification tasks, using its corresponding output hidden state (C in Figure 2.3) as the aggregate sequence representation. Concerning [SEP], it is used to separate the tokens from

different sentences (remember that the input of BERT can be pairs of sentences).

Finally, the representation of the input tokens is the sum of the token embeddings, its segment (either if the token is at the left or the right of the special token [SEP]) and the positional embeddings. This procedure is shown in Figure 2.4. Note how token embeddings and position embeddings (and how they are combined) are not novel with respect to the Transformers proposal (section 2.2.3). The main differences are the introduction of the special tokens ([CLS] and [SEP]) and segment embeddings, which are necessary to handle the representation of pairs of sentences.

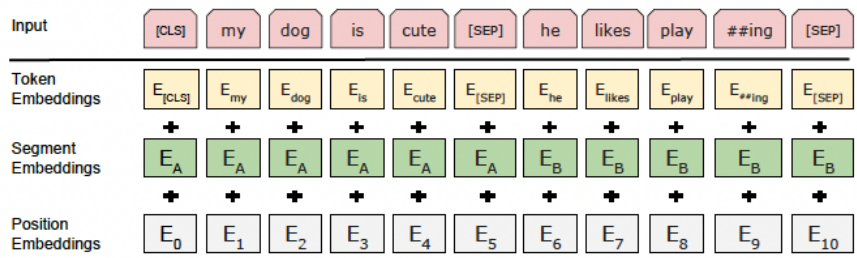

Figure 2.4: BERT input representation

2.3.2 Pre-training BERT

BERT is pre-trained on two unsupervised tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP). The pre-training corpora are BooksCorpus (800M words) and English Wikipedia (2500M words)

As previously explained, models like ELMo or GPT used unidirectional language models. They did that because, until that moment, it was not possible to implement bidirectional conditional language models without indirectly seeing the target word, making the task trivial. BERT overcomes this issue with a simple and clever alternative: if you don't want to see the target word, simply mask it. For that, 15% of the input tokens are randomly masked. The training objective is to predict the masked tokens. This task is called Masked LM (MLM) or Cloze. Note how for the pre-training, like language modelling is a token prediction task, the original BERT architecture needs to be modified with a fully connected output layer (with as many units as the vocabulary size) and a softmax activation function.

The other pre-training objective is NSP. NSP is introduced to ensure that the model is able to understand the relationship between sentences, helping to improve the performance on tasks that work with pairs of sentences (e.g. NLI). The task itself is quite simple, given a pair of input sentences A and B, 50% of the time B is going to be the actual next sentence after A. The objective is to predict if B is the sentence after A or not. As this is a sentence-level task, only the hidden state corresponding to the special token [CLS] is going to be used. Again, the architecture is modified adding some additional fully connected layers and a softmax activation function with cross-entropy loss.

2.3.3 Fine-tuning BERT

As said, the self-attention mechanism in the Transformer model is ideal to model many NLP tasks. Thus, it is easy to fine-tune BERT (indeed is pointed out as one of its most important strengths by the authors). Actually, it is only necessary to adapt the inputs to the task and add some task-specific output layers. Afterwards, the parameters of the model have to be fine-tuned end-to-end.

If the task is at token-level, the output layers have to be fed with the final hidden states corresponding to the subword tokens (T in Figure 2.3). On the contrary, if it is a sentence-level task, the output layers have to be fed with the final hidden state associated with the special token [CLS] (C in Figure 2.3).

In this Master Thesis, the task to perform is NERC. Therefore, the fine-tuning configuration that needs to be implemented is the one associated with token-level tasks. In that regard, the output layer for the NERC task can be either a Fully Connected or CRF layer [47]. In the following subsection, we present a brief recap of linear-chain CRFs.

2.3.3.1 Linear-chain CRF

The model form of a linear-chain CRF is as follows:

$$P(y|x;w) = \frac{exp(\sum_{i=1}^{n} wf(x,i,y_{i-1},y_i))}{\sum_{z \in Y} exp(\sum_{i=1}^{n} wf(x,i,z_{i-1},z_i))}$$

Where x is a sequence of words, y is the corresponding sequence of tags, f are feature vectors (in this case, the output of BERT) and w are the parameters of the

model. In inference mode, the goal is to compute the sequence of tags y that maximizes the equation above, i.e. $\underset{y}{argmax} P(y|x;w)$. This can be solved with the Viterbi algorithm. Finally, to find w that best fits the training data (parameter estimation), it is necessary to maximize the conditional log-likelihood of the training data.

2.4 Unsupervised Cross-Lingual LM

The transformer-based language models that were able to push the limits in natural language understanding (NLU) have been extended to more languages in order to deal with Cross-Lingual Understanding (XLU) tasks, in which a model is learned in one language and applied to other languages. See how this matches the long-term objectives of ieConsumo, i.e. having the ability to deal with multilingual data (Chapter 1). In the following subsections some of the most relevant cross-lingual language models are going to be presented.

2.4.1 Multilingual BERT

As explained in section 2.3.2, BERT is pre-trained on English corpus. To adapt to XLU tasks and to have the ability to handle multiple languages at the same time, the authors of BERT proposed multilingual BERT (mBERT) [8]. It is worth noting that, to my knowledge, they didn't publish any paper presenting the approach. The readme on their github repository is the only official source of information [8].

mBERT is nothing but an extension of BERT in the multilingual space. Indeed, the architecture and pre-training objectives are exactly the same in both approaches. However, mBERT presents some specific considerations that are worth mentioning. To begin with, it is pre-trained on the top 104 languages with the largest Wikipedias. Nevertheless, there is a lot of size variation among them. Thus, if the sampling strategy is proportional to the corpus size, low-resource languages may be under-represented (used fewer times during pre-training, ergo fewer resources allocated to them). To compensate for this, the authors introduced an exponentially smoothed weighting sampling strategy, i.e. sampling after exponentiating the probability of each language (proportional to its corpus size) by some factor and then normalizing it. The factor they decided to use is 0.7, which means that high-resource languages are under-sampled and low-resource languages are over-sampled. For tokenization, it still uses a shared WordPiece vocabulary, but

increasing its size from 30k to 110k (necessary to handle the increase of the number of words due to the inclusion of more languages). Concerning fine-tuning, the authors claim that it does not require any special consideration with respect to standard BERT.

The available mBERT models use $BERT_{base}$ configuration, i.e. $L = 12, H = 768, A = 12$. Note that, with the increase of the WordPiece vocabulary size from 30k to 110k, the number of parameters of the models go from 110M to 172M.

2.4.2 XLM-R

XLM-RoBERTa (XLM-R) [12] is a transformer-based multilingual masked language model pre-trained on text in 100 languages. Prior unsupervised cross-lingual language models like mBERT [8] or XLM [28] (same authors as XLM-R) were able to push the performance to achieve the state-of-the-art in several XLU tasks. In addition, they did it despite working on a quite small scale for both model complexity (total number of parameters) and training data (Wikipedia in their case). Taking this into consideration, XLM-R follows an analogous reasoning to the one followed in the monolingual space by RoBERTa [13]: try to improve the performance by pre-training longer and on more data. Following this approach, RoBERTa proved that BERT was undertrained. Similarly, XLM-R shows that mBERT and XLM are undertuned, obtaining their approach considerably better results.

XLM-R changes the pre-training strategy of BERT and mBERT and focuses only on MLM, without including NSP. During the pre-training, in every iteration, each batch of data belongs to a single language. For that, it is necessary to sample streams of text from each language. The authors proposed to follow the sampling distribution that they described in [28], which is a multinomial distribution with probabilities q_i where:

$$q_i = \frac{p_i^\alpha}{\sum_{j=1}^{N} p_j^\alpha} \quad \text{with} \quad p_i = \frac{n_i}{\sum_{k=1}^{N} n_k}$$

Note how α is a variable that controls the exponential smoothing of the language sampling rate. With $\alpha = 1$ the sampling distribution is simply proportional to the size of the training corpus. This can be harmful to lower-resource languages, as the model will be biased towards the languages with more data. Introducing α alleviates this issue. Note how this sampling strategy is exactly the same as the

one presented in mBERT (but properly formalized in a paper instead of a readme). Despite mBERT uses $\alpha = 0.7$ and XLM uses $\alpha = 0.5$, XLM-R goes with $\alpha = 0.3$, which means that it is giving even more relevance to low-resource languages than its predecessors.

Like most of the previous approaches for both monolingual and multilingual transformer-based language models, the authors proposed to use sub-word tokenization. In this case, they did it directly on raw text data using Sentence Piece [26] (instead of WordPiece) with a unigram language model [25]. The vocabulary size for this sub-word representation is large in comparison to the other exposed methods, going up to 250k units (remember that BERT has 30k units and mBERT 110k units).

Concerning the pre-training data, analogously to the logic applied in RoBERTa (which uses 10 times more data than the original BERT implementation), the authors aimed to improve the performance of the state-of-the-art by scaling up the amount of data. Concretely, instead of using Wikipedia, they built a CommonCrawl corpus in 100 languages (2.5 TB of data). In their approach, not only did they increase the overall dataset size but also they focused on increasing the amount of data for the lower-resource languages. Figure 2.5 shows the difference in size (GB) between CommonCrawl and Wikipedia for each of the training languages. Note how there is a notorious difference in all the languages but especially in the ones with fewer resources in Wikipedia.

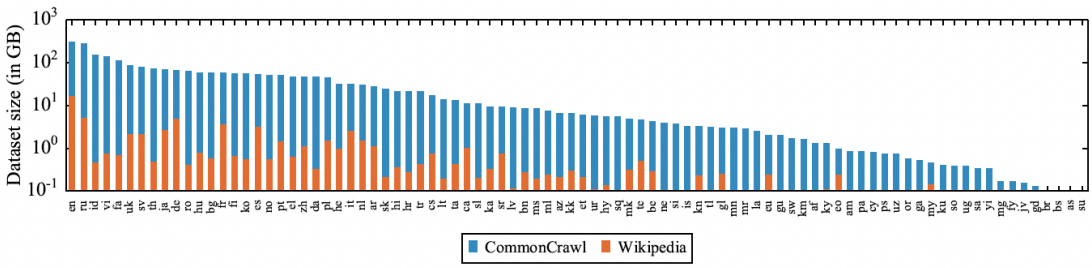

Figure 2.5: Wikipedia vs. CommonCrawl Dataset sizes

The authors presented two different pre-trained XLM-R models:

- $XLM\text{-}R_{base} : L = 12, H = 768, A = 12$. Replicates mBERT. 270M parameters.
- $XLM\text{-}R_{large} : L = 24, H = 1024, A = 16$. 550M parameters.

XLM-R consistently outperforms mBERT on the tested XLU tasks (XNLI, NERC and QA). For both XNLI and QA, XLM-R sets new state-of-the-art results. Concerning our task of interest, NERC, the results are on pair with previous state-of-the-art (LSTM + CRF approach described in [29]), improving mBERT by 2.41pp on F1 score on cross-lingual transfer (fine-tuning only in English) and 1.86pp on F1 score after fine-tuning on each language.

2.5 Data Augmentation for NLP

In NLP (like in most AI fields) the quality of the system is highly dependent on the amount of available data. For example, in Neural Machine Translation, the worst results are usually from low-resource language pairs, where the lack of data leads to poor quality. In most of the cases getting more data is not even an option as it is expensive and time-consuming. Therefore, data augmentation (i.e. generating additional synthetic data from the available data) arises as a suitable solution [32]. Indeed, the usage of data augmentation for deep learning in image processing has become a standard technique, but it has not been so common in NLP tasks. In the following subsections some possible approaches to augment textual datasets are going to be presented.

2.5.1 Easy Data Augmentation

Easy data augmentation [51][32] is probably the simpler approach to augment textual datasets. It is a process where a set of simple operations are applied to generate new synthetical data. For example, random swap consists in randomly selecting n words in the sentence and swapping their positions to generate new sentences. Another option is to perform random deletion or insertion of words. Finally, the most interesting technique is synonym replacement, which consists in replacing a randomly selected word by its synonym. In section 2.5.3 more concrete ways on how to handle this process are going to be presented.

2.5.2 Back-Translation

Back-Translation [43], is a technique that was firstly introduced in the context of Neural Machine Translation (MT). In this scenario (i.e. with a dataset of pairs of sentences in source and target languages), the idea is that by translating from the target language to the source language, the dataset may be increased by generating

some small differences from the original source sentence. This technique proved to be especially useful in pairs of languages with low-resources. This idea can be extended from MT to other tasks. However, like the datasets are no longer going to be pairs of source-target sentences, it is necessary to add an additional translation step. Thus, the sentences are translated to a pivot auxiliary language and then are translated back to the original language. In the process, while preserving the semantics of the sentence, some words may change, augmenting the dataset. To ensure that the dataset is sufficiently augmented, it is useful to carefully select the pivot language so that it does not share a lot of vocabulary and structure with the source language [20] [32].

2.5.3 Word Replacement

A logical yet powerful way to augment a textual dataset is to replace words with plausible alternatives. Contrary to easy data augmentation, this technique ensures that the new sentences make sense. In the context of MT, the authors of [30] presented an approach that consisted in targeting low-frequency words (referred as rare words) and creating new synthetical contexts containing them. For that, they looked for contexts in which a high-frequency word could be replaced by a rare word. Note that their objective was not to preserve the original meaning of the sentence but to ensure that the new pair maintained the semantic equivalence while being plausible (i.e. fluent and grammatically correct). For a given high-frequency word, the substitution was done by selecting the K words in the rare-word vocabulary with the highest conditional probability according to its left and right context. For that, they used forward and backward LSTM Language Models [54]. The final replacements were decided by selecting the rare words that were in the top K substitutions for both the forward and backward context. It is important to note that this paper was written before the Transformers era. For this reason, it was necessary to use two unidirectional LMs (forward and backward) to bidirectionally cover the context. Since BERT, this is no longer necessary as it can properly handle bidirectionality (thanks to the Masked LM pre-training task). After the publication of [30], the idea of augmenting the dataset by smartly replacing some words, i.e. taking into consideration their context, has gained popularity. Nowadays, with the explosion of the transformer-based models, a common and effective technique is to use synonym replacement using/comparing contextualized word embeddings (hidden representations from the input subword tokens) from BERT or RoBERTa (instead of the LSTM LMs used before).

Chapter 3

Methodology

3.1 Data

As explained in the Introduction (Chapter 1), the available raw data for the project comes from clinical courses mostly from Hospital de Nens de Barcelona and Hospital de la Vall d'Hebron. Such courses were specifically written to be as informative as possible of the circumstances that triggered the visit to the doctor, with a special focus on the products involved in the harmful event. For that, the medical doctors were requested to write the courses following an adaptation of the procedures defined by the Consumer Product Safety Commission (CPSC) of the United States in the National Electronic Injury Surveillance System (NEISS) [5], whose main purpose is "to collect data on consumer product-related injuries occurring in the United States". The analysts of CPSC use NEISS data to check the number of injuries associated with particular products and to detect potential hazards. The application of NEISS in the catalan health system was conducted by ieConsumo with its PIPADI program [22]. As in the case of CPSC and NEISS, ieConsumo aims to use PIPADI to detect emerging hazards and performing risk evaluations of consumer products.

When a hospital works with either NEISS or PIPADI, the medical records are coded according to their Coding Manual. Among the coded variables, an anonymized narrative description of the incident is included. These descriptions were the raw data to be annotated to perform the NERC task. The raw PIPADI dataset includes 6487 descriptions. Below you can find an example of one of them:

Se ha tragado el tapón de plástico de la botella de coca-cola. Estaba en casa. Tiene una erosión de 3mmm en el pilar derecho (esofago).

To obtain meaningful information from raw descriptions, it is necessary to define what is the relevant information for the task. In the concrete case of NERC, the labels for the named entities need to be specified. These are the labels that were decided according to the needs of the company:

- Product (PRD): Consumer good involved in the incident.
- Part of Product (POP): Part of the consumer good involved in the incident.
- Harm (HARM): Injury.
- Part of Body (POB): Part of body injured.
- Inductor (IND): Action/moment from which the normal course of the activity is changed, leading to a harming mechanism (without a defect or operating failure of the product involved).
- Trigger (TRG): Defect or operating failure of the product, leading to the incident.
- Harming Mechanism 1 (HM1): The step just before HM2 that irretrievably lead to HM2.
- Harming Mechanism 2 (HM2): The step that produces the injury or damage.

The labels were defined after several discussions and with the objective of meaningfully sequencing the harmful incident. Given that our goal was to analyse risks associated with consumer products, PRD and POP are essential tags: if there is not a consumer product involved, the description cannot be included in the final dataset. Excluding the mandatory entity, the most important tags are IND, TRG, HM1 and HM2. These tags help the analyst understand the context of the incident and why it happened. It is impossible to sequence the harmful incident without these tags. Most specifically, TRG is probably the most decisive tag, because an operating failure or defect should raise all the alarms when evaluating the risk of a product. Finally, HARM and POB give an idea of the harmful consequences of the incident and its severity. These tags can help to prioritize the evaluation of products involved in incidents with more severe consequences.

3.2 Annotation

After defining the entity labels, the next step was to annotate the PIPADI raw dataset. As explained in section 2.1, there are several free annotating tools that can be suitable for this task: Brat [31], Doccano [11] and INCEpTION [9]. Brat has not been updated since 2012 and it does not look like it is being maintained. INCEpTION is up-to-date but it is quite complex to use. Finally, Doccano looks like a good balance: is maintained and is more straightforward than INCEpTION. However, regardless of the selected tool, all of them present a huge handicap: they require some tech expertise, from Docker containers to Python or Java. In this work, an important requirement was that the annotation could potentially be done by people without advanced technical knowledge. To adapt to the business needs, an in-house annotation tool was built: PAT (PIPADI Annotator Tool).

It is important to clarify that it is not the objective of PAT to compete with the aforementioned tools, which are better and more scalable for an expert user in the field. PAT is simply a convenient solution to give answer to the company requirements:

- A tool that is able to support named entity annotation.
- A tool that can be used by annotators with all types of background.
- A tool that does not require the installation of programming languages.
- A tool that supports collaborative annotation.
- A tool with version control.

Analysing the requirements, Google Sheets [18] was selected for building the UI of PAT. Note how by default Google Sheets already support the last four requirements. It was part of this project to handle the first (and most important) one. Figure 3.1 shows the UI of PAT. The spreadsheet has a first column with the clinical course description (raw PIPADI dataset) and afterwards as many columns as the labels defined in section 3.1. The annotator, after reading the description, should *exactly* transcript the entity in the text in its corresponding column. If some description has more than one entity of a certain label, the different entities should be separated by adding ";" between them.

PAT - Mastersheet								
Raw Data	Product	Part of Product	Harm	Part of Body	Inductor	Trigger	HM1	HM2
Estaba jugando y se pillo los dedos con la puerta de entrada. Estaba en casa. Le han hecho una radiografia. Tiene un traumatismo en el dedo de la mano. El tratamiento es sutura por cirugia	puerta		traumatismo	dedos; dedo de la mano				se pillo
Quería salir de la bañera y ha caido de cabeza. Estaba en el baño de casa.Tiene una herida incisa superficial en el mentón. Lo han eviado a casa después del tratamiento de sutura con dermabond + steri-streeps.	bañera		herida incisa superficial	cabeza; mentón	Quería salir		ha caido	

Figure 3.1: PIPADI Annotation Tool (PAT) - Mastersheet

The PAT spreadsheet is connected with a Python backend that checks the annotated and validated sentences and, if they do not present transcription errors, tokenizes them, generates its BIO labels (see section 2.1) and adds them to the final dataset. Afterwards, the PAT mastersheet receives the feedback from the internal process. If a sentence has been correctly added to the dataset, its status is set to *Done*. On the contrary, if the process is not successful due to transcription errors, the annotator receives feedback on the words that have been incorrectly transcribed. This way, the errors can be corrected faster, and a better UX is given to the annotator. For more details about the tool and concrete annotation guidelines, you can check the Annotation Manual available in Appendix A.

The final PIPADI dataset was annotated by a single annotator (myself). It is stored in a *json* file following the format of CoNLL-2002 [48] and CoNLL-2003 [49]. Table 3.1 shows an example of the dataset.

Token	**NER tag**
Se	B-HM2
pillo	I-HM2
los	O
dedos	B-POB
con	O
la	O
puerta	B-PRD

Table 3.1: PIPADI dataset example

The PIPADI dataset has a total of 3501 annotated sentences. See how the original raw dataset (i.e. only the raw descriptions of the clinical courses) is larger, with 6487 sentences. The mismatch comes from the fact that not all the descriptions are suitable for the PIPADI dataset. This happens when there is no consumer product in the description of the event. In such case, the sentence has no interest, as it is not possible to perform a risk evaluation from it, and is filtered out. See an example of this situation in Table 3.2.

Description	**Included in dataset**
La bicicleta se ha roto y se ha caido al suelo	✓
Ha tropezado y se ha caido al suelo	X (no consumer good involved)

Table 3.2: Sentences to include in the dataset

Concerning the distribution of labels in the dataset, Table 3.3 shows the number of sentences containing at least one entity per each of the tags and the number of entities and tokens available per tag.

	PRD	**POP**	**HARM**	**POB**	**IND**	**TRG**	**HM1**	**HM2**
Nº sentences	3498	570	3365	3276	1540	405	1582	1353
Nº instances	5388	625	6485	6184	1823	437	1677	1445
Nº tokens	7837	866	11514	11616	3648	907	3261	3056

Table 3.3: Distribution of labels in PIPADI dataset

From Table 3.3 it can be concluded that almost all the sentences in the dataset have at least one entity of the labels PRD, HARM and POB. On the other hand, there are two clear minority classes, POP and TRG. In section 3.4 some actions to handle the class imbalance in the dataset are going to be presented. In addition, it can be seen how in terms of both instances and tokens, HARM and POB are clearly the majority classes. This makes sense because in most of the clinical courses the patient presents multiple injuries in several parts of the body, while the products involved (PRD) and the sequence of actions leading to the injury (IND, TRG, HM1 and HM2) are usually single instances.

Finally, during the annotation process, it could be observed how most of the clinical courses present spelling mistakes and grammatical errors. After generating the PIPADI dataset, a more concrete analysis was conducted and it was stated that 95% of the sentences present at least one error. With concern about this big error

rate and with articles like [27] reporting the bad operation of Transformer-Based models in noisy scenarios, a corrected PIPADI dataset was also generated. The grammatical corrector of Google Docs [17] was used for such a purpose. Note that during the correction process the original number of tokens and their ordering were not modified (we needed to have a human-in-the-loop for that). This was done to take advantage of the available annotations, to speed up the process. In the results section (Chapter 5), the performance using the original and corrected dataset is compared.

3.3 Modelling

This section presents the models and methods used to perform the NERC task from our self-annotated datasets. The strategy consisted in generating a baseline using a Rule-Based system and afterwards improving it by fine-tuning pre-trained multilingual Transformer-Based models.

3.3.1 Rule-Based System

When thinking about Artificial Intelligence most people think about Deep Learning and very few about rule-based systems. However, in the industry, they are still one of the preferred methods. Among their virtues, there is one that stands out: explainability. Contrary to deep learning black-box models, rules are easy to interpret and to handle even for a non-technical audience. For a company like ieConsumo, with the most experienced professionals in the field but with a lack of technological background, this is an important asset. Therefore, it was worth it to implement a rule-based system, at least to have a baseline for the task.

Another advantage of using a rule-based system in this project was that, given that we self-annotated the dataset, we gathered a lot of knowledge and spotted patterns of the dataset during the annotation process. To be fair, as we collected information from the entire dataset during the annotation (and not only from the training set), our rules may be a little bit biased. Thus, it is possible that we suffer from a little bit of data leakage. Apart from that, the data in the validation and test set was not used to generate the rules. All the resources and insights came exclusively from the data in the training set (following the dataset split explained in Chapter 4)

The final rule-based system is composed of 32 rules. Most of them are based on patterns learnt during the annotation process. They combine typical structures, focusing on the presence of certain keywords and the part-of-speech (POS) of the words surrounding them, with resources extracted from the training dataset. Such resources are basically dictionaries of the most common entities per each of the tags. The ordering is very relevant in a rule-based system: if a token satisfies more than one rule, the one that appears in the first place is the one that is going to be applied. Thus, the first rules are the ones that represent the most common structures in the dataset. For example, a very common structure in PIPADI is the one where the doctors write down their prognosis: *Tiene un HARM en el POB*. In this case, the keywords would be *Tiene* and *en*. The words in their surrounding with *NOUN* as POS would be tagged as *HARM* and *POB*, respectively. In the last rules, the dictionaries are applied. This way, if a token does not meet the typical structures, the prediction will be the label with which is commonly tagged. Finally, it should be taken into consideration that at this stage the objective was not to create very complex rules but to have a system that could provide a sufficiently good baseline. If you want to check the final set of rules, you have them available in Appendix B.

3.3.2 Transformer-Based Model

The main approach developed for this project consisted in fine-tuning a pre-trained multilingual transformer-based model for the NERC task. This section goes through the process of selecting the model to implement. Each of the subsections presents a set of discussions to get to the final proposal. Note that all the experimentation referred to in this section is conveniently explained in Chapters 4 and 5.

3.3.2.1 mBERT vs. XLM-R

Given that the dataset presented in section 3.1 is in Spanish, the first decision to be made was whether to use a model solely pre-trained in Spanish (like BETO [3]) or go for a more general multilingual approach. Despite BETO outperforms by 1pp the best multilingual BERT model in the Spanish corpus for NERC task, we decided to go for a multilingual model. The reason is that, even though for this project the data is exclusively in Spanish, the company aims to use a similar model to manage complaints from users from all over the world. Thus, the ability to handle different languages with a single pre-trained model makes the multilingual

approach very convenient for us.

Nowadays, the referent multilingual transformer-based models are mBERT [8] (see section 2.4.1) and XLM-R [12] (see section 2.4.2). Although XLM-R outperforms mBERT in the NERC task by a relevant margin (1.86pp), not only performance should be taken into consideration. XLM-R was born with the idea to prove that mBERT was undertrained and for that, it was pre-trained for longer and with more data and higher model complexity. Therefore, there is a trade-off between performance and computational cost. Although it is true that most of the increase of computational cost is handled during the pre-training phase, it also has some impact during fine-tuning (more parameters to fine-tune). Table 3.4 (taken from [12]) shows a comparison of their model architectures and sizes.

Model	**#lgs**	**tokenization**	**L**	H_m	H_{ff}	**A**	**V**	**#params**
mBERT	104	WordPiece	12	768	3072	12	110K	172M
XLM-R	100	SPM	24	1024	4096	16	250K	550M

Table 3.4: Multilingual model architectures and sizes. *#lgs* are the number of pre-training languages, L is the number of layers, H_m is the number of hidden states of the model, H_{ff} is the dimension of the feed-foward layer, A the number of attention heads and V is the size of the sub-word vocabulary.

With the context provided by table 3.4, the trade-off between mBERT and XLM-R in the NER task is an increase of 1.86pp at the cost of having to fine-tune almost 400M extra parameters. Given the scarcity of computational resources (more on that in Chapter 4) and the limited time available for this project, mBERT looks like a more convenient solution. Thus, it was the model selected for the project.

3.3.2.2 DistilmBERT vs. mBERT

DistilBERT [38] is a distilled version of BERT. Its authors generated it by applying knowledge distillation during the pre-training phase of BERT. Their model reduces the size of BERT by 40% while maintaining 97% of its capabilities. In addition, the reduction of size significantly speeds up the training (60% faster). Such characteristics are ideal for projects with little computational resources, like ours. Fortunately, mBERT has also its distilled version (DistilmBERT), which is twice as fast as mBERT. It has 6 layers, dimension of 768 and 12 attention-heads (134M parameters vs. 172M in mBERT).

The strategy followed in this project consisted in trying to optimize as much as possible the computational resources. Therefore, all the experimentation and hyperparameter-tuning were done with DistilmBERT. In the experimentation phase, using DistilmBERT instead of BERT allowed us to perform more experiments in the same amount of time, which in the end resulted in a more complete and better training. As a final step, to try to push the performance, mBERT can always be used with the best DistilmBERT configurations.

3.3.2.3 DistilmBERT for NERC

DistilmBERT, like mBERT, has the same architecture as BERT. Therefore, as explained in section 2.3.3, we needed to add some output modifications to adapt it for the NERC task. For this task, the fine-tuning configuration that needs to be implemented is the one associated with token-level tasks, i.e. relying on the hidden representation of each of the input tokens instead of the one corresponding to the special token [CLS] (used for sentence classification tasks). For that, it is necessary to add an output layer on top of the original architecture. In most of the experimentation done the selected output was a single fully-connected layer with a softmax activation and cross-entropy loss function. In addition, like several articles like [12] or [39] show the benefits of using an output CRF layer (with a negative log-likelihood loss function), this approach was also tested.

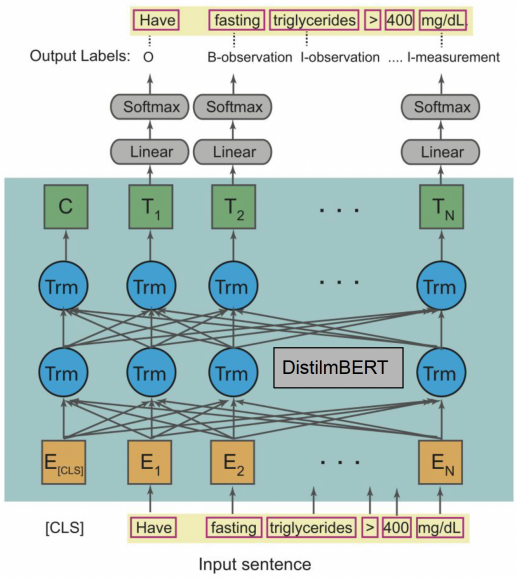

Figure 3.2: DistilmBERT-Linear architecture for NERC [4]. The architecture with CRF is equivalent. Only needs to substitute the Linear layer by a CRF layer.

3.4 Data Augmentation

As shown in section 3.2 the final PIPADI dataset has 3501 annotated sentences, which looks enough for fine-tuning DistilmBERT. However, having a little bit more data, especially containing the minority classes POP and TRG (remember Table 3.3), could help to improve the performance and compensate the class imbalance of the dataset.

Easy Data Augmentation is one of the Data Augmentation techniques for NLP included in the Literature Review (section 2.5). Despite its simplicity (or precisely because of that), some of their proposed techniques (like random swapping/deletion) could introduce noise to the dataset while lacking explainability, e.g. it may be difficult to justify to a non-expert audience why a random swap helps to improve the performance. For this reason, we implemented more elaborated data augmentation techniques, like Back-Translation and Word Replacement using Contextualized Word Embeddings. In the following subsections, details of the approach followed for both of them are presented.

3.4.1 Back-Translation

In Back-Translation [43], the idea is to augment the dataset by modifying some words of a sentence in the process of translating it to a pivot language and then going back to the original language. There are a couple of important decisions to be made when using Back-Translation in a dataset. First, the selection of the aforementioned pivot language, which is very important. Ideally, the pivot should be as different as possible from the original language, this way there are more chances to augment the dataset. Second, how to do the translations.

When applied to this project, we selected English as our pivot language. Therefore, we did a *Spanish-English-Spanish* back-translation process. Here, it is true that the recommendation to choose a very different pivot language is not 100% fulfilled, there are languages that are much more different from Spanish than English. However, it was done out of convenience: one thing is adding small modifications to the dataset and another one is to add a lot of noise. This effect was reduced when using English rather than other options. Concerning the translator, although a good and logical option was to use the python's free library Googletrans [15] (that implements Google Translate API), the selected option was Marian [24]. The

main reason is that Googletrans has a maximum request limit while Marian (open source) can manage as many requests as necessary.

The whole training data was back-translated using the Marian NMT model (with a human-in-the-loop checking that the generated sentences made sense). The issue with this approach is that it can modify the ordering of the words and the number of tokens in the sentence. This was not desirable because it prevented us from taking advantage of the annotations done for the original dataset. Consequently, after back-translating, it was necessary to annotate again. However, it was not feasible at that stage to re-annotate all the training set. Thus, it was done exclusively for a sample of the training data (398 sentences). Note how this augmented dataset does not modify the distribution of classes shown in Table 3.3, as no actions were taken to balance the dataset when using this technique.

3.4.2 Word Replacement with CWE

As seen in section 2.5.3, [30] proposes a way to increase the dataset by smartly replacing some words. In their proposal, they performed the substitution of words using forward and backward LSTM LMs. Now, their approach can be replicated with the representations obtained from bidirectional transformer-based models.

To implement this technique in the project, several decisions had to be taken. 1. Define the target entities, 2. How to obtain the Contextualized Word Embeddings (CWE), 3. Evaluate similarity between the different CWE, 4. Implement the changes to the dataset. You have an overview of the whole process in Figure 3.3.

3.4.2.1 Define target entities

In [30], the focus is to improve the translations involving rare-words, generating new synthetic contexts for them. In our case, the issues came from an imbalance in the dataset, with a couple of labels (POP and TRG) clearly underrepresented, and a label (IND) difficult to annotate. Replicating the idea in [30], the goal was to generate new contexts for the entities of these labels. To do that, we substituted from the sentence entities belonging to other classes. Concretely, from PRD and POB, which are present in almost all the sentences, increasing the likelihood of augmenting the dataset for the target entities. Let's clarify the idea with an example:

La silla ***se ha roto*** *y el niño ha caído. Tiene una fractura en la clavícula.*

In the example, in bold we have the target entity and underlined the possible entities to substitute: *se ha roto* belongs to TRG, *silla* is the PRD and *clavícula* is the POB. See how, for example, *silla* could be substituted for another product like *bicicleta*. With that, we would generate a plausible substitution that increases the examples for TRG. Note how PRD and POB are not only convenient because they are very common entities but also because it is easier to substitute them and get logical substitutions. Other entities, like HM1 or HM2, are usually verbs, which are more complex to substitute without compromising the plausibility. Finally, the decision to substitute more than one type of entity is aligned with the findings described in [30], where they obtained better results when performing more than one substitution.

3.4.2.2 Obtain CWE

According to the approach shown in [40], we obtained the Contextualized Word Embeddings (CWE) from the final DistilmBERT layer (to be consistent with 3.3.2). Note that we used the pre-trained model directly in inference mode, i.e. before fine-tuning it with our dataset. The embeddings allowed us to substitute words for others with similar meanings. Therefore, we needed to generate them for the entities that could be potentially substituted, PRD and POB. Then, to perform the final substitution, these embeddings were compared with the embeddings of different entities belonging to the same label. Moreover, to check the influence that the context has on the quality of the embedding, we also generated context-free embeddings for these entities. In this case, instead of inputting whole sentences to DistilmBERT, we employed PRD and POB entities in isolation (check Figure 3.3). In principle, context-based embeddings should be better. However, the extra variability added by the context-free approach can also lead to promising results. In the results section (Chapter 5), both approaches are compared.

3.4.2.3 Similarity between CWE

In the PIPADI dataset it is quite common to have multi-token entities. Given that the embeddings were generated at token level, it was necessary to find a way to represent them at entity level. To solve this issue, the embeddings from tokens belonging to the same entity were simply averaged. Once we had them, we needed

a way to evaluate the possible substitutions. For that, we used cosine distance [52]. To ensure a certain level of quality and generate new sentences as plausible as possible, only the embeddings with the distance below a certain threshold were actually substituted. These thresholds, that we determined experimentally, were set to 0.2 and 0.3 for the context-based and context-free embeddings, respectively. To perform the final substitution both the replacements for PRD and POB had to ensure the quality constraint; otherwise, they were discarded.

3.4.2.4 Implementing changes in dataset

An important advantage of this approach with respect to Back Translation is that we do not rely on external systems to implement it: the embeddings are generated from DistilmBERT and the rest of the process has been implemented by ourselves. This is important because using the black-box MarianMT model for Back-Translation prevented us from taking advantage of the annotations done for the original dataset, forcing us to re-annotate the new sentences. On the contrary, with this approach, we have absolute control of the process of substitution, which allows us to automatically generate the new labels from the original annotations. As a result, almost all the training dataset can be augmented. In Chapter 4 different configurations for the augmented dataset (i.e. with more or less augmented sentences) are compared.

3.4.3 Oversampling

Finally, as one of the reasons to do Data Augmentation is to compensate the class imbalance, we also performed a typical strategy to deal with this situation: oversampling the minority class. In this case, we experimented with different configurations in which sentences containing the conflictive labels (TRG, POB and IND) were duplicated.

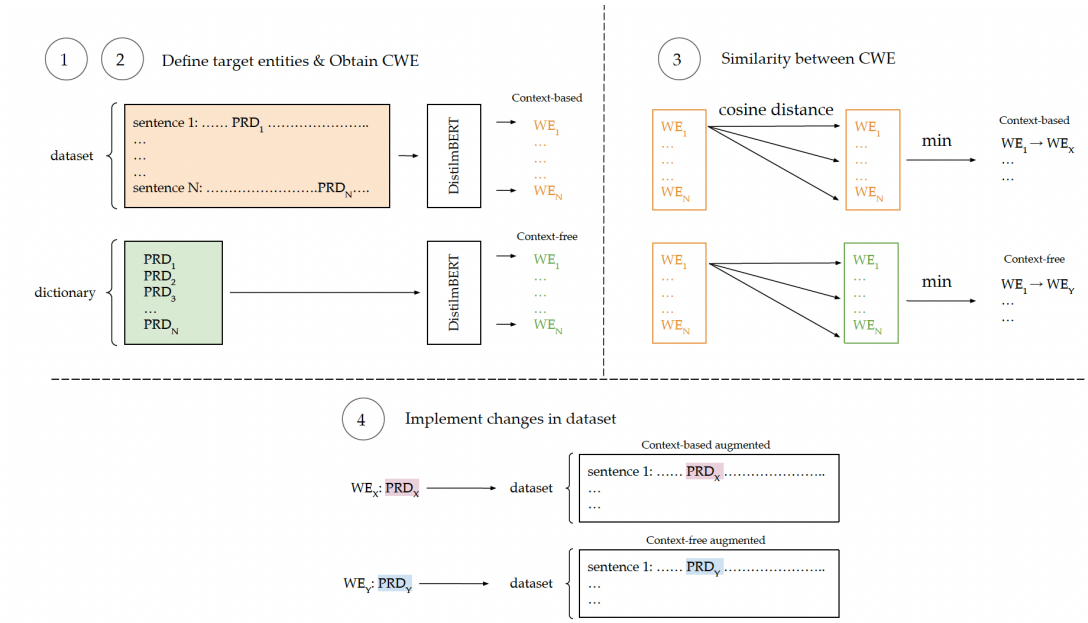

Figure 3.3: Overview of Data Augmentation with CWE. The first action to make was to select which entities we wanted to replace (step 1). In our project were PRD and POB. In the figure, we exemplify it with PRD. There were two options: either computing the embeddings from the PRD entities in their context (context-based approach, in orange) or computing them in isolation, from a dictionary only containing PRD (context-free approach, in green). For both options, we got the embeddings (WE in the figure) from the last layer of DistilmBERT (T green boxes in Figure 3.2) (step 2). The embeddings are vectors of dimension 768. To compare them (step 3), we used cosine distance. In the context-based approach, we compared the context-based embeddings with other context-based embeddings (from different sentences). On the contrary, in the context-free approach, we compared context-based embeddings with context-free embeddings (green box). In both cases, for each embedding, we selected as its replacement the embedding that is at the minimum distance and ensures a certain level of quality. Finally (step 4), we generated the two augmented datasets replacing the original entities with the ones obtained in the previous step.

Chapter 4

Experiments

4.1 Experimental setup

For the rule-based system (section 3.3.1) the only particularity of the implementation is the way to obtain the POS tags for Spanish tokens. We used the Spanish pipeline `es_core_news_sm` from spaCy [46]. Bear in mind that we used this system only as a baseline. The core of the experimentation consisted in fine-tuning a pre-trained DistilmBERT model. To implement it, we used the PyTorch `transformers` library from Hugging Face [58][14]. Most concretely, the `distilbert-base-multilingual-cased` model ($L = 6, H = 768, A = 12$). The following subsections deepen on the requirements and particularities to fine-tune it.

4.1.1 Input formatting

When fine-tuning BERT (or mBERT), there are input formatting requirements that need to be satisfied (remember section 2.2.3). To begin with, the input must be tokenized and we need to add the special tokens [CLS] and [SEP] at the start and end of each sentence. Then, all of them are converted to their ID in the pre-trained model vocabulary. The whole process can be done with the tokenizer `DistilBertTokenizerFast` from the `transformers` library. Nevertheless, like our dataset was already tokenized, we directly added the special tokens and mapped them to their IDs. When analysing the token IDs, we realized how a surprisingly high number of them were set to 100, which is the ID corresponding to the special token [UNK]. When this happens means that these tokens were not present in the vocabulary used to pre-train the model. We expected this to happen in the presence of typos but not for words perfectly written, which was the case. As seen, DistilmBERT tokenizer (WordPiece) can break words into several sub-words.

Hence, commonly seen sub-words can also be represented by the model [60]. In our case, even if the original word is out-of-vocabulary, it is likely that their sub-words are not, which solves the issue. Consequently, what we did was to apply `DistilBertTokenizerFast` on top of the tokens from the dataset. When doing so, we also had to align to a sub-token-level the token-level labels.

After tokenization, the resulting length of the sentences may vary a lot. To have a uniform input format, all sentences need to be padded or truncated to a fixed length, `MAX_LEN`. When padding, the special token [PAD] is added to the sequence. In our dataset, the maximum length of the training sentences (after WordPiece tokenization) is 121. To give some margin for potential longer sentences in the test set, we set `MAX_LEN=220` (still far from BERT's maximum sentence length, 512 tokens). The [PAD] tokens are mapped to their token ID, 0. Finally, it is necessary to generate the attention masks which is simply an array of 1's and 0's used to indicate which are the [PAD] tokens. This mask indicates to the self-attention mechanism to ignore the [PAD] tokens in its calculations.

Note that both padding and the generation of the attention masks can also be handled by `DistilBertTokenizerFast`. In our implementation we applied it to perform the entire input formatting process: sub-word tokenization, padding, generation of attention mask and mapping of the tokens to the ID in the model vocabulary.

4.1.2 DistilmBERT for NERC

We needed to modify the pre-trained DistilmBERT model to perform the NERC task. For that, we employed the interface `DistilBertForTokenClassification` from Hugging Face. This implementation allows to easily load the pre-trained model and adds the necessary output layers to perform the NERC task. Concretely, it is simply the DistilmBERT architecture with a token classification head (linear layer) on top of the hidden-states output. Although almost all the experimentation was done with this DistilmBERT-Linear model, we also tried to see if it was useful or not to use a CRF layer (DistilmBERT-CRF) instead. In [45], they obtained their best results with this model head. Unfortunately, Hugging Face does not provide an interface for BERT-CRF. To do the implementation, we used as reference the code published by [45] in [44]. For both model heads, we set a dropout rate of 10% after DistilmBERT's output.

4.1.3 Parameter setting

Following the fine-tuning recommendations given by the authors of BERT (Appendix A.3 in [7]), for the hyperparameter tuning task we exclusively focused on the following options:

- Batch size: 16, 32
- Learning Rate: 2e-5, 3e-5, 5e-5
- Epochs: 2, 3, 4

For the optimizer, Adam was used (`AdamW` class from the `transformers` library). Apart from the learning rate, it is also required to set the value for Adam's epsilon parameter. For that, we used the default value `eps=1e-8`.

4.1.4 Data

For the experimentation and evaluation of the project, we divided the PIPADI dataset (and its corrected version) following an 80-10-10 (train-devel-test) split. The split was stratified on the minority classes (POP and TRG) to ensure that their proportions were preserved. The final splits were saved to ensure reproducibility. The tables below summarize the characteristics per each partition.

	Nº sentences	**Nº tokens**
Training	2800	34100
Devel	350	4337
Test	351	4268

Table 4.1: Volumes of the dataset splits

	PRD	**POP**	**HARM**	**POB**	**IND**	**TRG**	**HM1**	**HM2**
Training	6311	683	9196	9237	2881	731	2636	2425
Devel	773	89	1165	1203	385	88	315	319
Test	753	94	1153	1176	382	88	310	312

Table 4.2: Number of tokens per label in each of the splits

In Tables 4.1 and 4.2 we can see how, effectively, the splits were correctly stratified, maintaining the proportion of the minority classes. As explained in section 3.4, to try to mitigate the imbalance issue, we implemented several data augmentation techniques, which focused on increasing the proportion of the minority class

while providing some more training examples. Apart from Back-Translation, with the approaches based on CWE and oversampling, we specifically augmented sentences where there were instances of the problematic labels. Remember how for CWE we wanted to compare the quality of the datasets generated through context-based embeddings (*CWE Free* in Tables 4.3 and 4.4) versus the ones obtained with context-free embeddings (*CWE Context* in Tables 4.3 and 4.4). In addition to the minority classes POP and TRG, we expected to have issues with the prediction of the label IND (it was even complex for us to annotate it). Consequently, in some of the approaches (denoted with + *IND*), we did not only focus on augmenting the minority classes but also the label IND, to a maximum of 750 augmented sentences. The information for each of the augmentation approaches is summarized in the tables below.

	Nº sentences	**Nº tokens**
Back-Translation	398	4357
CWE Free	525	11437
CWE Free + IND	750	15936
CWE Context	600	9580
CWE Context + IND	750	11808
Oversampling	525	7668
Oversampling + IND	750	10698

Table 4.3: Volumes of the augmented datasets

	PRD	**POP**	**HARM**	**POB**	**IND**	**TRG**	**HM1**	**HM2**
Back-Translation	822	104	1038	1438	272	65	266	352
CWE Free	2416	481	2023	4608	586	507	302	514
CWE Free + IND	3446	481	2723	6455	1069	507	523	732
CWE Context	2019	553	2214	2560	712	563	347	612
CWE Context + IND	2519	553	2687	3210	1050	563	479	747
Oversampling	1362	481	2023	1893	586	507	302	514
Oversampling + IND	1933	481	2792	2651	1112	507	539	683

Table 4.4: Number of tokens per label in the augmented datasets

In Table 4.4 it can be seen how, apart from Back-Translation, all the approaches augmented the proportion of tokens belonging to the minority classes. In addition, the + *IND* approaches did likewise for the label IND. The approach followed for CWE augmentation consisted in replacing entities from classes PRD and POB

from the sentences containing difficult labels (full explanation in section 3.4.2.2). Hence, it is interesting to check the variation in the number of tokens corresponding to these labels. The training dataset has on average 2.25 PRD tokens and 3.3 POB tokens per sentence. These values go up to 4.6 and 8.7 for *CWE Free* and to 3.3 and 4.3 for *CWE Context*. The fact that these values increase makes sense, as in a lot of cases the replacement is done by adding some adjectives to the original entity. However, the rise looks much more reasonable for the context-based approach. Concerning the context-free approach, the replacements, despite making sense, are sometimes a little bit weirder, which can explain the huge increase. This situation is exemplified in Table 4.5. Apart from that, the sentences containing the difficult labels are usually longer than the rest, which proportionally increases the token count.

	Sentence
Original	Mientras andaba se golpeo contra el **armario**
CWE Context	Mientras andaba se golpeo contra el **armario de madera**
CWE Free	Mientras andaba se golpeo contra el **armario de telefonica roto**

Table 4.5: In the Original row, the entity to replace is highlighted in bold. It can be seen how with *CWE Context* replacement it is substituted by *armario de madera* while in *CWE Free* is substituted by *armario de telefonica roto*. Both replacement entities exist in the dataset. Although both solutions are correct and make sense, the one from *CWE Context* looks more natural. This proves that the quality of the replacement increases when the embeddings take into consideration the context.

4.1.5 Hardware

All the fine-tuning experimentation is executed on Google Colaboratory [16], taking advantage of their GPUs (NVIDIA Tesla K80 [33]) accessible for free.

4.2 Evaluation Metrics

For the evaluation, we used the standard metric for NERC task: F1-score [21]. F1-score is the harmonic mean of the Precision and Recall metrics.

$$Precision = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

$$Recall = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision + Recall}}$$

Precision and Recall are inherently binary metrics (either the prediction is a positive or a negative). For this reason, when the data is multiclass it is necessary to define how to compute them. One option is using `micro` F1-score, which calculates the metrics globally, i.e. counting the total number of true positives, false negatives and false positives. The other option is `macro` F1-score, which is the unweighted mean of the metrics for each label. In this project, we decided to use `macro` F1-score. The reason is that some of the most relevant labels are underrepresented. Thus, using `micro` F1-score, even with very bad results in these labels the metric would look good. On the contrary, evaluating with the `macro` strategy gives a better idea of the real/practical quality of the model.

As said, the PIPADI dataset was labelled using the BIO format. Hence, given that there are 8 type of entities (PRD, POP, HARM, POB, IND, TRG, HM1 and HM2) plus the output label (O), we had a total of 17 unique classes. Despite the predictions following the BIO format, i.e. predicting per each token its tag with `B-` or `I-`, we decided to do the evaluation following an IO format. This was done because it is really uncommon in our dataset to have consecutive entities from the same label, while it is more frequent for the model to correctly predict the entity but fail between `B-` and `I-`. In addition, we are not reporting the results including the class O and we ignore the predictions for the special tokens [CLS], [SEP] and [PAD], as they are not interesting for the task.

Chapter 5

Results and Discussion

In this Chapter, the most relevant results are presented. Apart from what is shown in this section, you can find the outcome for our whole experimentation in Appendix C.

5.1 Baseline

The first step in the experimentation was to obtain a baseline using our rule-based system (section 3.3.1). Table 5.1 shows the results obtained in the devel set.

	Total	**PRD**	**POP**	**HARM**	**POB**	**IND**	**TRG**	**HM1**	**HM2**
F1-score	**58.52**	63.91	18.52	88.69	79.95	19.53	54.24	75.62	67.91

Table 5.1: Results in the devel set with the rule-based system.

The macro F1-score using the rule-based system gets to 58.52%. It is interesting, though, to check the results at entity level. As expected, the classes identified as most difficult (POP, IND, TRG) are the ones getting the worst results. However, it is interesting to highlight how in TRG the result is quite good (54.24%). This happens because TRG is a class that does not have overlapping tokens with other labels: their entities are quite exclusive. Therefore, it is relatively easy to get correct results simply by checking if the token is present in the dictionary of most typical entities from the training set. Something similar happens with PRD, HM1 and HM2. Finally, the labels getting the best performance are HARM and POB. This is something expected as these entities always appear following the same structure in the descriptions of the dataset. Remember that you can have a closer look at the final set of rules on Appendix B.

5.2 Original vs. Corrected Dataset

In section 3.2 it has been explained how, due to the great number of clinical courses containing typos and mistakes, we created the corrected version of the PIPADI dataset. We wanted to verify if, like it is exposed in [27], the performance would be compromised by the presence of misspelling errors. Thus, we tested the performance of fine-tuning a DistilmBERT-Linear model on the original and corrected datasets (Table 5.2). Note that this was done prior to a proper process of hyperparameter tuning. For this experiment we used `Epochs=4, Batch Size=32, LR=2e-5`.

	Total	**PRD**	**POP**	**HARM**	**POB**	**IND**	**TRG**	**HM1**	**HM2**
Original	**83.38**	89.36	72.21	96.93	97.61	69.14	63.53	87.20	91.01
Corrected	**83.84**	89.92	74.48	96.79	97.54	69.87	63.75	87.74	90.64

Table 5.2: Comparison in the devel set between original and corrected datasets.

Table 5.2 shows how the corrected dataset outperforms the original one. However, the difference is smaller than expected (0.46pp). This is explained by the fact that we applied WordPiece tokenization. As explained in 4.1.1, by splitting incorrect words, it is more likely that their sub-words are present in the model vocabulary, which makes BERT-based models robust to errors. Given the similar performance, the costs associated with the corrected approach should be taken into consideration. To begin with, if we trained our models with corrected sentences, we would have to ensure that the testing data (i.e. the real use case) is also corrected. Therefore, the company would have to include a good and automatic corrector as a previous step to apply their service. In our opinion, it is not worth it to add this (potentially expensive) extra step for such a small difference in performance. For this reason, the corrected dataset was discarded and we performed the rest of the experimentation solely on the original dataset. Apart from that, it is impressive to see how, just with a first trial configuration, it is possible to massively improve the results of the baseline (around 25pp). At entity level, the model keeps struggling with the difficult labels POP, IND and TRG. However, in the case of POP and IND, the improvement is huge: 56pp and 50pp, respectively. On the other hand, TRG is the label that improves the less (9pp).

5.3 Hyperparameter Tuning

The core of the experimentation consisted in fine-tuning a pre-trained DistilmBERT-Linear model. For that, we performed hyperparameter tuning according to the recommendations given for BERT (section 4.1.3). Apart from that, we tested the data augmentation techniques implemented in this work. Figure 5.1 compares multiple combinations of hyperparameters and augmentation techniques. For the sake of clarity, experiments with the same combination are averaged.

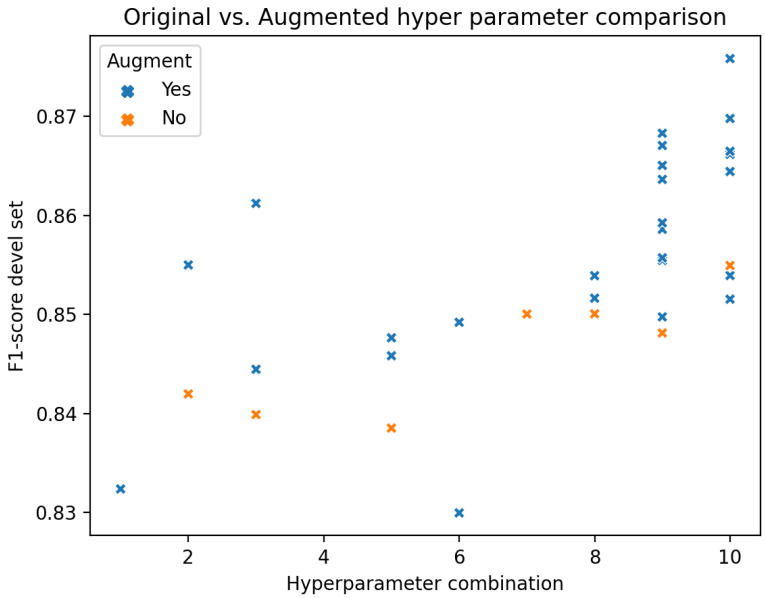

Figure 5.1: Comparison of models with original vs. augmented datasets. The hyperparameter combination legend can be found in Table 5.3

	Num. Epochs	Batch Size	Learning Rate
Combination 1	2	16	2e-5
Combination 2	4	16	2e-5
Combination 3	3	16	2e-5
Combination 4	5	32	2e-5
Combination 5	4	32	2e-5
Combination 6	2	32	5e-5
Combination 7	4	32	5e-5
Combination 8	3	32	5e-5
Combination 9	2	16	5e-5
Combination 10	3	16	5e-5

Table 5.3: Legend of Hyperparameter combinations.

In Figure 5.1 it can be seen how, regardless of the configuration, the performance on the devel set is better when fine-tuning the model with an augmented dataset (blue markers). As you can observe, there are more blue (augmented) than orange (non-augmented) markers. The reason why this happens is because there were a lot of data augmentation configurations to test (remember Table 4.4), while the non-augmented was just one. In addition, you can also note how Combinations 9 and 10 have much more samples than the rest. This is due to the fact that in our initial experiments we spotted that with these configurations the performance was better. Hence, we tried more data augmentation options on them. On Figure 5.2 we can check on detail each of them. For the sake of clarity, experiments with the same combination are averaged.

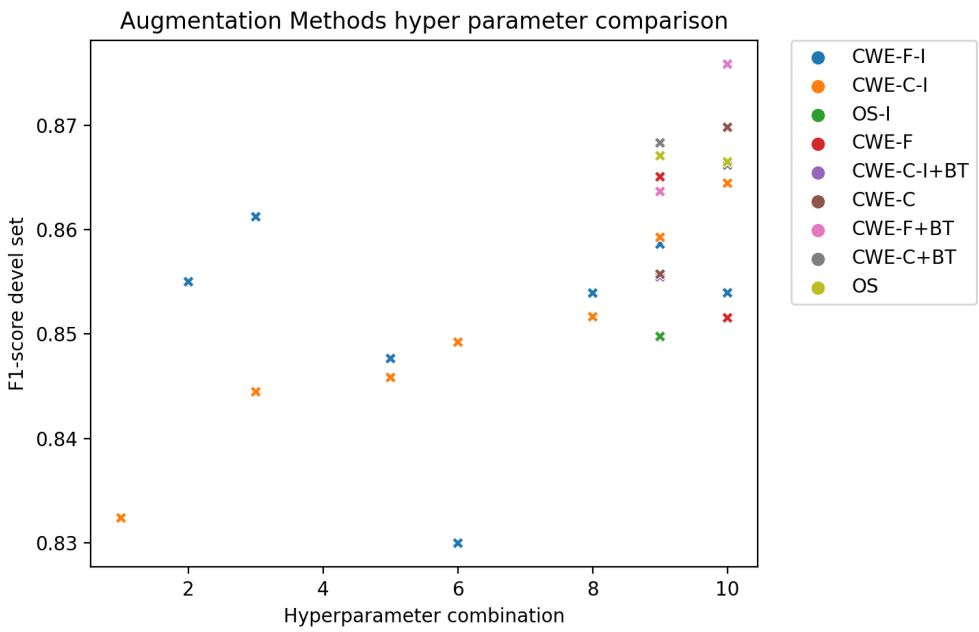

Figure 5.2: Comparison of models with different augmented datasets. It is the replica of Figure 5.1 without the non-augmented dataset and specifying the augmentation strategy followed in each case. To relate the legend with the explanation in 4.1.4 and Table 5.3: *CWE-F* stands for *CWE Free*, *CWE-F-I* for *CWE Free - IND*, *CWE-F+BT* is the combination of *CWE Free* and *Back-Translation*, *CWE-C* for *CWE Context*, *CWE-C+BT* is the combination of *CWE Context* and *Back-Translation*, *CWE-C-I* for *CWE Context - IND*, *CWE-C-I+BT* is the combination of *CWE Context - IND* and *Back-Translation*, *OS* for *Oversampling* and *OS-I* for *Oversampling - IND*

With data augmentation, the initial approach was to directly handle the complex labels of the dataset. For this reason, the original techniques were *CWE Free - IND* (blue marker) and *CWE Context - IND* (orange marker), which generated

new contexts for sentences containing the difficult labels (TRG, POP and IND). It can be seen how their performance (especially for *CWE Context - IND* - orange marker) is progressively growing with the successive hyperparameter combinations. When reaching *Combination 9*, there was a relevant peak in performance and we decided that, from there, we would test new configurations. At that point, *CWE Free - IND* and *CWE Context - IND* were successfully improving the performance on TRG and POP, but were failing to rise it up for IND. We noticed that when generating new contexts for sentences with TRG and POP, we were already adding contexts for IND, as these labels tend to coincide in the same sentence. Thus, the contexts specifically generated for IND were not strictly necessary. Indeed, they were overfitting the model for that label. To solve this issue, we decided to get rid of them and we created *CWE Free* and *CWE Context*, focusing exclusively on TRG and POP. Moreover, we decided to include also the sentences generated through Back-Translation (*CWE-F+BT* and *CWE-C+BT*), as they were not biased to any class and provided a good way to increase the variability on the training set. Finally, we compared our approaches to the effect of simply oversampling training examples including the difficult labels TRG and POP *OS* and also IND *OS-I*.

As seen in Figure 5.2, the new data augmentation approaches boost the performance on the devel set. However, like *Combination 9* is only trained for 2 epochs (without severe overfitting), we were concerned to be wasting the full potential of the models. Hence, we increased the number of epochs to 3 (*Combination 10*). With this combination, despite the models overfitting a little bit more, the performance increases (in some cases). The final leading configurations belong exclusively to these combinations (top right corner in Figure 5.2).

To sum up, the main findings from Figures 5.1 and 5.2 are the following:

- Employing an augmented dataset improves the performance.
- The augmentation should be exclusively focused on the minority classes (TRG and POP). Despite IND being a complex class, providing more contexts for this label worsens the performance, as the model overfits the training data.
- There is not a very relevant difference in performance between augmenting with context-based (*CWE-C*) rather than context-free (*CWE-F*) embeddings. Depending on the configuration, both of them can perform well. Indeed,

among the best models both options are present (*CWE-F+BT* vs. *CWE-C* and *CWE-C+BT*).

- The combination of word replacement and Back-Translation enriches the augmented dataset to the point that 2 of the top models use them (*CWE-F+BT* and *CWE-C+BT*).
- Classical approaches to deal with the imbalance datasets (Oversampling) are an easy, fast and inexpensive approach to boost the performance.

As seen in section 4.1.2, as a last step for the hyperparameter tuning process, we also tried changing the model head to a CRF layer (DistilmBERT-CRF). Taking advantage of our findings with a Linear head, we directly applied the top configurations to the new model. Table 5.4 shows the best results with this approach.

Data Augmentation	**Epochs**	**Batch Size**	**Learning Rate**	**Macro F1**
Non-Augmented	10	16	5e-5	**75.86**
CWE-F+BT	8	16	5e-5	**78.94**

Table 5.4: Best results with DistilmBERT-CRF on devel set.

The results shown in Table 5.4 are not up to the expectations that we had for DistilmBERT-CRF, being around 10pp below the equivalent configurations with DistilmBERT-Linear. The main limitation comes from the complex classes, where low results are obtained. Data Augmentation seems to help in that sense (+3pp), but it is not enough to ensure sufficiently good quality. However, this does not mean that this model is not a useful approach; simply, it probably requires much more dedication to the fine-tuning process than what we devoted. Actually, we didn't do it because of our restrictions on computational resources. As it can be seen, fine-tuning this model requires much more epochs than with a Linear layer. In addition, the time per epoch increases from 120 to 355 seconds. In practical terms, this means that in the time necessary to fine-tune DistilmBERT-CRF once, around 10 experiments could be done with DistilmBERT-Linear. And this is not only a matter of time but also that long executions penalize you on Google Colaboratory. When we experimented with DistilmBERT-CRF, we usually surpassed the free GPU usage limit and we had to wait for around 24 hours to continue the experimentation. All in all, under these circumstances, it was not worth it for us to devote more time to this model.

5.4 Final results

This section summarizes the performance of our best models on the devel set (Table 5.5) and shows the final results on the test set.

	Total	PRD	POP	HRM	POB	IND	TRG	HM1	HM2
10: *CWE-F+BT*	**88.06**	90.69	84.68	97.59	97.80	73.29	76.97	90.25	93.18
10: *CWE-C+BT*	**87.44**	89.91	82.46	97.72	97.67	71.40	77.78	88.61	94.00
9: *CWE-C+BT*	**87.37**	90.41	84.58	97.49	97.59	72.41	76.12	88.17	92.16
10: *CWE-C*	**87.27**	90.82	80.15	97.84	97.55	70.74	78.79	89.67	92.62
10: *OS*	**87.15**	90.03	81.65	98.00	97.47	69.99	78.03	89.50	92.55
9: *CWE-F+BT*	**87.05**	90.11	76.81	97.48	97.50	71.32	82.84	88.26	92.12
10: *Non-Aug*	**86.06**	89.47	81.00	97.09	97.72	70.86	72.16	88.22	91.94
Baseline	**58.52**	63.91	18.52	88.69	79.95	19.53	54.24	75.62	67.91

Table 5.5: Results on the devel set with the best configurations. The baseline and the best results obtained without data augmentation are also included.

As you can check on Appendix C, there are a lot of configurations getting an F1-score over 86% on the devel set. In Table 5.5 we have decided to show only the ones above 87%. As we can see, among these top performers we have exclusively the hyperparameter combinations 9 and 10. Interestingly, without data augmentation, the best results obtained are not that far from the augmented versions (around 1-2pp). When it comes to the complex labels, apart from TRG, the non-augmented version obtains comparable results, especially on POP.

With the best configurations we evaluated on the test set:

	Total	PRD	POP	HRM	POB	IND	TRG	HM1	HM2
10: *CWE-C+BT*	**86.39**	89.68	71.98	98.13	98.26	75.38	75.43	89.21	93.18
10: *OS*	**85.71**	89.57	67.95	97.84	98.16	74.88	78.03	87.80	91.43
10: *CWE-F+BT*	**85.16**	89.11	68.00	97.98	98.01	73.76	75.51	86.98	91.91
10: *CWE-C*	**84.99**	89.85	67.87	98.18	98.37	73.71	72.16	87.58	92.23
9: *CWE-F+BT*	**84.96**	90.06	69.53	97.77	98.29	73.07	72.20	87.05	91.75
10: *Non-Aug*	**84.96**	89.07	71.48	98.13	98.57	74.86	69.03	87.46	91.04
9: *CWE-C+BT*	**84.27**	88.14	63.60	98.08	97.80	73.10	71.43	89.03	92.96

Table 5.6: Results on the test set with the best configurations.

5.5 Analysis of General Results

Table 5.6 shows the results on the test set. Some of the most important findings are that the top 3 results use *Combination 10*, which shows its superiority with respect to *Combination 9*. All the models decrease their performance with respect to the devel set, but some of them are particularly affected: *9: CWE-F+BT*, -2.09pp; *10: CWE-C*, -2.28pp; *10: CWE-F+BT*, -2.9pp; *9: CWE-C+BT*, -3.1pp. Indeed, only *10: CWE-C+BT*, *10: OS* and *10: Non-Aug* decrease less than 2pp. This shows how, in general, with the augmented datasets we tended to overfit to the devel set. This is especially important for the POP class, with only one of the augmented models (*10: CWE-C+BT*) being able to go over 70%, while on the devel set almost all of them surpassed the 80%. On the contrary, the augmentation proves to be very useful for the TRG label, consistently outperforming by a relevant margin (2-6pp) the results for the non-augmented model. When it comes to the rest of the classes, although quite similar, the performance is slightly better for the augmented approaches (especially for the top 2). Despite its strengths and weaknesses, the best performing models on the test set use augmentation techniques, outperforming by a relevant margin (1.43pp) the best non-augmented model. In any case, we should be more careful not to overfit the devel set when applying data augmentation in the future.

To conclude, *10: CWE-C+BT* is our best model. Performing a Wilcoxon test [57] at entity-level we proved that its performance is statistically significantly better (p-values < 0.05) than the other models, with the exception of *10: OS* (p-value = 0.148). This last point is particularly relevant, we spent a lot of resources and time working on both Word Replacement with CWE and Back-Translation. However, working with Oversampling was a matter of minutes. This approach is optimal in terms of the cost-benefit trade-off. On the one hand, it improves the performance with respect to the non-augmented approach. On the other hand, it does not require the dedication of the aforementioned data augmentation techniques. When it comes to the comparison between context-based (*CWE-C*) and context-free (*CWE-C*) replacement, the context-based approach has proved to perform better on the test set (context-free seems to overfit more the devel set). Finally, it is worth mentioning that data augmentation is not an exact science and that the best-performing approaches in this work may not be suitable for other datasets.

5.6 Analysis at entity level

Performing an analysis at entity level is key in our project. Not only does it help to evaluate the performance of our model but also reflects the quality of the annotation process. In fact, some of the results make much more sense from the annotator perspective. Using the information from Figure 5.3, we will give feedback for some of the entities:

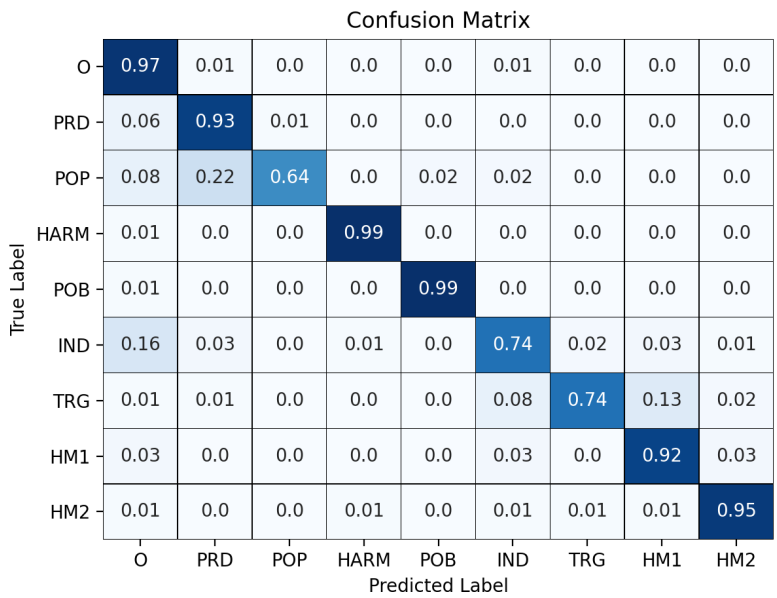

Figure 5.3: Normalised confusion matrix from best model (*10: CWE-C+BT*) on the test set.

- PRD: 6% of their entities are predicted as O. See how, from all the entities, PRD is the third with the highest confusion with O (only after two complex classes, POP and IND). We believe that this value is especially high in PRD because of how the annotation was done. PRD should not be a complex class to annotate, but if we remember the definition given in section 3.1, we should only care about products involved in a harmful event. Therefore, despite being in the sentence, all the products that are not involved in an incident have not been tagged as PRD. We believe that, although this decision makes sense at business level, in terms of the operation of the model is sub-optimal. Maybe a better solution would be to tag all the products in the sentence as PRD and, afterwards, perform some post-processing or apply some relation extraction model to determine whether it is involved in the incident or not.

- POP: Creating the POP label was probably a design mistake. Actually, we had doubts until the last moment. To begin with, POP is a difficult label to annotate. For example, it is not unlikely to find situations where there are parts of parts of product, which makes the annotation trickier. In addition, there are sentences in which the same token is PRD and other sentences where it is POP, e.g. *Se ha enganchado con la **puerta*** vs. *Se ha enganchado con la **puerta** del coche*, in the first case ***puerta*** would be clearly tagged as PRD and in the second case as POP. As a result, 22% of their tokens in the test set are predicted as PRD. We believe that a good option (especially given the low number of POP tokens) would be to annotate all POP as PRD and afterwards determine the relation (*part of*) by means of some relation extraction model. In that sense, we have tried using some simple rules that have achieved promising results (around 60% of F1-score for POP on devel).

- IND: The trickiest class to annotate. Probably its definition has been too broad. In it, we look for the moments right before the harmful event. However, within this definition, we can include a lot of different things, which makes it difficult to find patterns for the annotations of this label. Consequently, it is the class that most times is incorrectly predicted with O.

- TRG: Probably the most decisive, yet scarce, label. It includes entities that show defects or operating failures of the product. This label is key because spotting defects in a product is at the core of risk assessment. Correctly detecting these entities can be vital to raise the alarm and prevent incidents to happen. The fact that with augmentation techniques we are able to boost its performance is one of the best news of this project. Concerning its performance, it presents a lot of overlapping with HM1, which explains that 13% of their tokens are predicted with that label.

Chapter 6

Conclusions

In this work, we fine-tuned DistilmBERT to perform information extraction (NERC) from clinical courses. The goal was to automatically extract relevant information necessary to evaluate the risks associated with the products involved in a harmful incident. Taking advantage of the experience in the field of the professionals of ieConsumo S.L., we defined the named entity labels to extract from the text: Product (PRD), Part of Product (POP), Harm (HARM), Part of Body (POB), Inductor (IND), Trigger (TRG), Harming Mechanism 1 (HM1) and Harming Mechanism 2 (HM2). After that, we annotated the raw dataset using an in-house annotation tool built to satisfy our business requirements, i.e. to be user-friendly for people without a technical background. The final dataset, built following the fashion proposed by CoNLL-2002/2003, has a total of 3501 annotated sentences. Although this amount of data might be enough to perform the task, the dataset is clearly imbalanced, with some classes clearly under-represented (POP and TRG). To overcome this issue, we studied and implemented several data augmentation techniques for NLP, like Back-Translation (BT) or Word Replacement with Contextualized Word Embeddings (CWE), and more classical ways to deal with imbalance (oversampling). After building a baseline with a simple rule-based system, we performed an exhaustive hyperparameter-tuning process to fine-tune DistilmBERT for NERC. Although we tried with both a Linear and a CRF head, we did most of the experimentation with the Linear output, as it was more convenient given our computational resources. Our experiments prove the worthiness of data augmentation, outperforming the non-augmented strategy. Our best approach is a DistilmBERT-Linear model trained with an augmented dataset (CWE+BT). It obtains good results on the task (86.39% F1-score), ensuring a sufficiently good quality (>70%) for all the labels, regardless of their support.

Chapter 7

Future work

Despite all the presented work, there are still things to be done related to this project. For example, we left for future work the task of re-annotating the dataset. The current version was labeled by a single annotator (myself), which is sub-optimal. Doing a second annotation could be beneficial to prevent mistakes. In addition, we could correct some of the issues that we found related to the label definition, like considering removing the POP tag or annotating every single instance of PRD. In this scenario, it could also be interesting to annotate relations between entities (to enable relation extraction). Furthermore, to try to push the performance of the system, it would be ideal to have better computational resources: more powerful and with higher availability. The free version of Google Colaboratory, despite being a convenient solution, limited our experimentation (especially) due to its usage limits. Better hardware would allow us to try with more powerful models (like mBERT or XLM-R) and to do a better job when using a CRF model head. Finally, if ieConsumo reached an agreement to handle the user complaints from a multinational company, it would be necessary to replicate the procedure followed in this project with the new data. In addition, this new multilingual dataset, together with our decision to work with a multilingual model, would allow us to test the cross-lingual ability of our approach.

Appendix A

Annotation Manual

A.1 System Overview

We have created an in-house tool to perform the annotation task. The way to proceed has consisted in using a shared Google Sheet to ensure a collaborative (and version controlled) tool while providing a good UX for non-tech-savvy users. This spreadsheet is controlled by an automatic bot (PIPADI bot from now on), which is in charge of checking the validity of the annotated and validated entries and generating the dataset from them.

The spreadsheet is composed of several blocks:

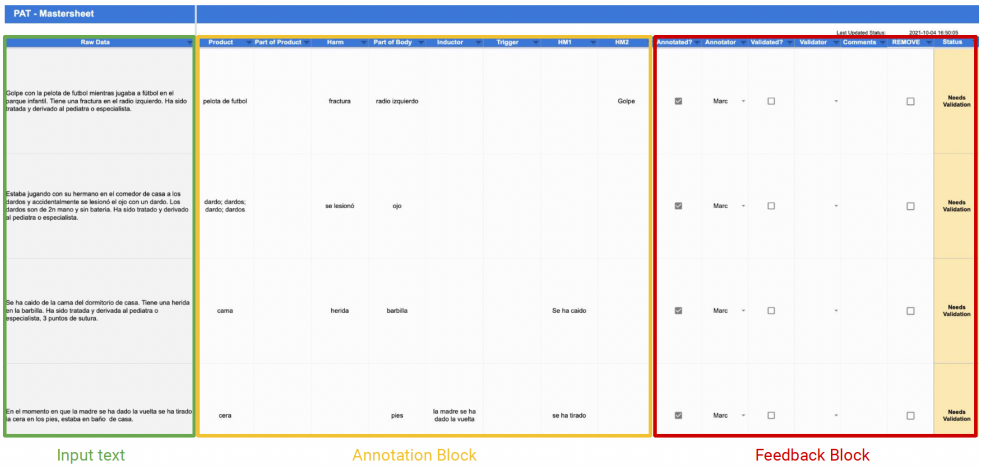

Figure A.1: PIPADI Annotation Tool - System Overview

- **Input Block**: Contains the raw text descriptions that have to be annotated. The user has nothing to do in that block.

- **Annotation Block**: Presents as many columns as the number of pre-specified labels to annotate the text with.
- **Feedback Block**: Presents information about:

  Who has labeled (and validated) a certain annotation.

  If the input entry should be removed.

  PIPADI bot feedback. The automatic bot will show you the annotation status of each input entry. Also shows the words incorrectly annotated in case of transcription errors.

A.2 General Annotation Guidelines

During the process of annotation, it is normal to face situations where the annotator may doubt between multiple options on how to label the text. It is for that reason that it is necessary to establish some clear guidelines to follow in these cases. In this section we show those guidelines that can be applied regardless of the label, as they are common for all of them:

- **Not annotating articles**: When annotating a noun, we have no interest in annotating the articles that go with it, as it does not include relevant information for the task.
- **Exactly transcribing the text**: include all the mistakes. We should prepare the machine to read spelling mistakes.
- **Annotating the same token as many times as it appears in the description**: if it appears 3 times, 3 times we need to annotate it (separating each annotation with ";").
- **Separate elements in enumerations of the same label**: do not include the connectors in the annotation.
- **Maintain order of appearance inside each label**: the order itself is not relevant when generating the dataset but will make things easier for the validator

Examples:

Case	Descripton	Annotation
Not annotating articles	Se ha golpeado con la **pelota**	pelota (PRD)
Exactly transcripting the text (including mistakes)	Se ha golpeado con un **valon**	valon (PRD)
Annotating multiple references to a token	Se ha golpeado en la **cara**. Tiene una contusión en la **cara**.	cara; cara (POB)
Separate elements in enumerations of the same label	Se ha golpeado la **cara** y el **brazo**	cara; brazo (POB)

Table A.1: General Annotation guidelines.

As a final reminder, it is essential to annotate all the references to the same token. Otherwise, we would be fooling the machine. It is also very important to correctly separate the multiplicities or enumerations of tokens with ";". In case this is not done, the annotations won't be validated.

In the following sections, some specific details about the annotation of each of the different labels are going to be presented.

A.3 Annotating the product

The product must be a consumer good, i.e. something that can be bought (not built) by a person or company. Thus, typical elements involved in incidents such as "escaleras " (stairs), "escalones" (steps), "pared" (wall) or "suelo" (floor), which are not consumer goods, should not be annotated as products.

Now, considering that a consumer good is present in the description, we can encounter several situations in which it may be complex to discern what is the exact span of the product. Let's list 4 typical complex scenarios that can happen when annotating a product:

- **The product is complemented (contextualized)**: Usually, a product can be the center of a Noun Phrase (sintagma nominal) which presents other phrases complimenting it:

Adjectives: may provide relevant information about the product that may help to contextualize it and to perform the hazard evaluation.

Example 1: "el niño cogió una pieza *pequeña*".

The fact that it is a small part is relevant, as it implies the possibility that the kid swallows it.

Example 2: "el niño chocó con una puerta *metálica*"

The material of the product is also relevant as it can affect the type and severity of the injury.

It is true that there are other adjectives that are completely irrelevant for the task, such as the ones related to the colour: "cogió el juguete amarillo". In any case, these kinds of details are not usually present in the description and therefore we include the adjectives in the product annotations.

Prepositional Phrases:

In Spanish, these phrases will typically start with the preposition "de", which leads to three different types of phrases:

Specifying the location

Example: "Se ha chocado con la puerta *de la entrada*"

The location where the incident takes place is not relevant. Therefore, should not be included in the annotation.

Specifying the material

Example: "Se ha chocado con la puerta *de cristal*"

The material of the product is also relevant as it can affect the type and severity of the injury. Therefore, should be included in the annotation.

Specifying the activity

Example: "Le ha golpeado la pelota *de fútbol*"

The activity of the product is relevant as it can affect the type and severity of the injury. For example, it is not the same to be hit by a ping-pong ball than a basketball ball. Therefore, should be included in the annotation

- **There are multiple products in the description**:

  Example: "Tropieza con la muleta, se choca contra la puerta de cristal y se corta el brazo"

In order to make things easier to model, both products (remember that should be consumer goods) should be annotated

- **Hidden products inside Harming Mechanisms** :

  Sometimes in Spanish there are products which are hidden inside a hidden mechanism. For example: "pedrada" (piedra), "martillazo" (martillo), "pelotazo" (pelota) or "balonazo" (balón)

  It would be a good practice to annotate these cases as products instead of harming mechanisms. Although it may be more complex for the model to understand, there is an underlying product that should be caught up. Moreover, once the information is extracted, these tokens can be easily converted into the product noun by means of the process of lemmatization.

A.4 Annotating: Inductor vs. Trigger

Checking the definitions of the proposed labels, it is clear how both Inductor and Trigger reflect a change in the normal course of an activity which finally leads to a harming mechanism (i.e. a harming mechanism can be neither Inductor nor Trigger). However, it is important to clarify the differences between them so as to ensure the correct annotation of entries. Concretely, the difference that the annotator should look for is if this change of the course of the activity is led by a defect or operating failure of the product involved. If this is the case, it should be labeled as Trigger; otherwise, as Inductor. It is important to note that this does not mean that the Inductor cannot be caused by a product, simply means that the product has behaved as expected.

Case	**Descripton**	**Annotation**
An inductor is never a harming mechanism	Iba corriendo, **se cayó** y se golpeó en el codo	se cayó (HM)
Inductor without a product involved	Iba corriendo, **se tropezó**, se cayó y se golpeó en el codo	se tropezó (IND)
Inductor with a product involved	**Al pisar** la pelota cayó y se golpeó en el codo.	Al pisar (IND)
Typical trigger: defect of product	La pelota **explotó** y le golpeó en el codo	explotó (TRG)

Table A.2: IND vs. TRG annotation guidelines.

Unlike most of the labels, that usually have kind of a predetermined structure, the Inductor is the label that can present a more free structure. Sometimes the action from which the event turns into a harmful incident can be kind of complex and we still should annotate it correctly. For example: "La madre dejó el paquete de pastillas y la niña se tragó todo el bote". Both "paquete de pastillas" and "bote" should be labeled as products, "se tragó" should be annotated as HM1 and finally "La madre dejó" is the Inductor as this is the action from which the action leads to the harmful event.

A.5 Annotating: HM2 vs. Harm

The harm mechanism may be composed of only one step or two linked steps. The last step of these two-linked steps (HM2) is the one that produces the injury or damage (Harm). Therefore HM2 is often confused with the Harm. For example, a knife cuts a person's skin resulting in a cut. It can lead to confusion: "cuts" is a verb, the harming mechanism, and "cut" is a noun, the harm itself, so be aware of this linguistic peculiarity.

In addition, when it comes to the Harm there are a couple of details that are worth mentioning. Firstly, the harm is often contiguous in the text to the severity associated with it. It is important to catch it so as to give more information about the injury for posterior analysis. Finally, it is common to see incidents in which the description reports more than one harm, remember in these cases to correctly annotate them, using ";" as separation.

Case	**Descripton**	**Annotation**
HM2 is a verb	**Le ha golpeado** la cara	Le ha golpeado (HM2)
Harm is a noun	Tiene un **golpe** en la cara	golpe (HARM)
Harm including contiguous severity	Tiene una **quemadura de 3º grado** en la mano	quemadura de 3º grado (HARM)
Harm including non-contiguous severity	Tiene una **quemadura** en la mano de **3º grado**	quemadura; 3º grado (HARM)
Enumeration of harms	Tiene un **golpe** en la cara y una **quemadura** en la mano	golpe; quemadura (HARM)

Table A.3: HM2 vs. HARM annotation guidelines.

A.6 Annotating: Part of body (POB)

This is probably one of the most straightforward labels to annotate. However, it is necessary to pay attention to a couple of situations that could lead to mistakes. On the one hand, this is probably the label that will appear more times inside a text due to the fact that it usually goes together with the Harming Mechanisms (HM2) and Harms. Remember to correctly annotate the enumerations and catch every single reference to a part of the body in the text, regardless you have already annotated it once for the same text.

Case	**Descripton**	**Annotation**
POB together with HM2	Se ha golpeado la **cara**	cara (POB)
POB together with Harm	Tiene un golpe en la **cara**	cara (POB)
POB together with Harm and HM2	Se ha golpeado la **cara**. Tiene un golpe en la **cara**.	cara; cara (POB)
Enumeration of POBs	Se ha golpeado en la **cara** y el **brazo**. Tiene una contusión en la **cara** y un esguince en el **brazo**	cara; brazo; cara; brazo (POB)

Table A.4: POB annotation guidelines.

Appendix B

Handcrafted Rules

```
import pandas as pd
import json
import numpy as np
import spacy
from sklearn.metrics import f1_score, confusion_matrix

# Load resources
HARM_RESOURCES = pd.read_csv('resources/dictionaries/pipadi/harm_resources.csv',
                             names=['resource', 'count'])['resource'].tolist()
HM1_RESOURCES = pd.read_csv('resources/dictionaries/pipadi/hm1_resources.csv',
                            names=['resource', 'count'])['resource'].tolist()
HM2_RESOURCES = pd.read_csv('resources/dictionaries/pipadi/hm2_resources.csv',
                            names=['resource', 'count'])['resource'].tolist()
IND_RESOURCES = pd.read_csv('resources/dictionaries/pipadi/inductor_resources.csv',
                            names=['resource', 'count'])['resource'].tolist()
POB_RESOURCES = pd.read_csv('resources/dictionaries/pipadi/part_of_body_resources.csv',
                            names=['resource', 'count'])['resource'].tolist()
POP_RESOURCES = pd.read_csv('resources/dictionaries/pipadi/part_of_product_resources.csv',
                            names=['resource', 'count'])['resource'].tolist()
PRD_RESOURCES = pd.read_csv('resources/dictionaries/pipadi/product_resources.csv',
                            names=['resource', 'count'])['resource'].tolist()
TRG_RESOURCES = pd.read_csv('resources/dictionaries/pipadi/trigger_resources.csv',
                            names=['resource', 'count'])['resource'].tolist()


def apply_rules(tokens, pos_tags):
    # Prepare list to store predictions from rule
    predictions = np.full(len(tokens), 'O').astype('<U16')
    for i in range(len(tokens)):
        if predictions[i] != 'O':
            continue

        if 0 < i < len(tokens) - 1:

            # Looking for typical Harm + POB pattern: Tiene HARM en POB.
            if tokens[i - 1] == 'Tiene':
                j, keep = 0, 1
                while keep == 1:
```

```
            if j == 0 and pos_tags[i + j] == 'DET':
                pass
            else:
                if tokens[i + j].lower() in POB_RESOURCES
                   and pos_tags[i + j] in ['NOUN', 'PROPN', 'ADJ']:
                    predictions[i + j] = 'POB'
                else:
                    predictions[i + j] = 'HARM'
            if pos_tags[i + j + 1] == 'PUNCT':
                keep = 0
            if tokens[i + j + 1] == 'en':
                keep = 0
                # Check for POB
                new_i = i + j + 2
                k, keep2 = 1, 1
                while keep2 == 1:
                    predictions[new_i + k] = 'POB'
                    if new_i + k == len(tokens) - 1:
                        break
                    if tokens[new_i + k + 1] == '.' or tokens[new_i + k + 1] == ',' \
                            or tokens[new_i + k + 1] == 'tambien'
                            or tokens[new_i + k + 1] == 'tambi n':
                        keep2 = 0
                    k += 1
                    if (k == 15 and keep2 == 1)
                       or (keep2 == 1 and new_i + k == len(tokens) - 1):
                        keep2 = 0
                        predictions[new_i + 1: new_i + k] = 'O'
            j += 1
            if (j == 15 and keep == 1) or (keep == 1 and i + j == len(tokens) - 1):
                keep = 0
                predictions[i + 1: i + j] = 'O'


        if tokens[i] == 'tambien' or tokens[i] == 'tambi n':
            init_z = max(0, i - 20)
            start_harm = 0
            for z in range(init_z, i):
                if tokens[z] == 'Tiene':
                    start_harm = 1
            if start_harm == 1:
                j, keep = 1, 1
                while keep == 1:
                    if j == 1 and pos_tags[i + j] == 'DET':
                        pass
                    else:
                        predictions[i + j] = 'HARM'
                    if pos_tags[i + j + 1] == 'PUNCT':
                        keep = 0
                    if tokens[i + j + 1] == 'en':
                        keep = 0
                        # Check for POB
                        new_i = i + j + 2
                        k, keep2 = 1, 1
```

```
                    while keep2 == 1:
                        predictions[new_i + k] = 'POB'
                        if new_i + k == len(tokens) - 1:
                            break
                        if tokens[new_i + k + 1] == '.' or tokens[new_i + k + 1] == ',':
                            keep2 = 0
                        k += 1
                        if (k == 15 and keep2 == 1)
                            or (keep2 == 1 and new_i + k == len(tokens) - 1):
                            keep2 = 0
                            predictions[new_i + 1: new_i + k] = 'O'
                j += 1
                if (j == 15 and keep == 1) or (keep == 1 and i + j == len(tokens) - 1):
                    keep = 0
                    predictions[i + 1: i + j] = 'O'

# Looking for typical "Quemaduras" gravity: extension/grado
elif tokens[i] == 'extension' or tokens[i] == 'extensi n':
    predictions[i] = 'HARM'
    j, keep = 1, 1
    while keep == 1:
        predictions[i + j] = 'HARM'
        if tokens[i + j] == '%':
            keep = 0

        if (j == 10 and keep == 1) or (keep == 1 and i + j == len(tokens) - 1):
            keep = 0
            predictions[i + 1: i + j + 1] = 'O'

        j += 1

elif (tokens[i] == '1  ' or tokens[i] == '2  ' or tokens[i] == '3  ') and \
        ('grado' in tokens[i + 1] or 'profund' in tokens[i + 1]
         or 'superf' in tokens[i + 1]):
    predictions[i] = 'HARM'
    predictions[i + 1] = 'HARM'
    if i + 2 < len(tokens) - 1:
        if 'profund' in tokens[i + 2] or 'superf' in tokens[i + 2]:
            predictions[i + 2] = 'HARM'

# Try to catch some POB which are not together
elif 'izquierd' in tokens[i] or 'derech' in tokens[i]:
    if pos_tags[i - 1] == 'NOUN':
        predictions[i] = 'POB'
        predictions[i - 1] = 'POB'

# Looking for product
elif tokens[i - 1] == 'contra' or tokens[i - 1] == 'con':

    if pos_tags[i] == 'NOUN' and tokens[i]
       not in ['suelo', 'pared', 'muro','escaleras',
               'escalon', 'extension', 'extensi n']:
        predictions[i] = 'PRD'
        if pos_tags[i + 1] == 'NOUN' or pos_tags[i + 1] == 'ADJ':
```

```
                predictions[i + 1] = 'PRD'
                if i + 3 < len(tokens):
                    if pos_tags[i + 2] == 'ADP' and pos_tags[i + 3] == 'NOUN':
                        predictions[i + 2] = 'PRD'
                        predictions[i + 3] = 'PRD'
            if i + 2 < len(tokens):
                if pos_tags[i + 1] == 'ADP' and pos_tags[i + 2] == 'NOUN':
                    predictions[i + 2] = 'PRD'
        if pos_tags[i] == 'DET' and pos_tags[i + 1] == 'NOUN' \
                and tokens[i + 1] not in ['suelo', 'pared', 'muro', 'escaleras',
                                          'escalon', 'extension', 'extensi n']:
            predictions[i + 1] = 'PRD'
            if i + 2 < len(tokens):
                if pos_tags[i + 2] == 'NOUN' or pos_tags[i + 2] == 'ADJ':
                    predictions[i + 2] = 'PRD'
                    if i + 4 < len(tokens):
                        if pos_tags[i + 3] == 'ADP' and pos_tags[i + 4] == 'NOUN':
                            predictions[i + 3] = 'PRD'
                            predictions[i + 4] = 'PRD'
            if i + 3 < len(tokens):
                if pos_tags[i + 2] == 'ADP' and pos_tags[i + 3] == 'NOUN':
                    predictions[i + 3] = 'PRD'

    elif tokens[i] == 'marca':
        if pos_tags[i + 1] in ['NOUN', 'PROPN']:
            predictions[i + 1] = 'PRD'
            if tokens[i - 1] == 'la' and tokens[i - 2] == 'de':
                if pos_tags[i - 3] == 'NOUN':
                    predictions[i - 3] = 'PRD'
                    j = 1
                    while pos_tags[i - 3 - j] in ['NOUN', 'ADJ', 'PROPN']:
                        predictions[i - 3 - j] = 'PRD'
                        j += 1
                        if i - 3 - j < 0:
                            break
                elif tokens[i - 3] == 'es' or tokens[i - 3] == 'son':
                    j = 1
                    while pos_tags[i - 3 - j] in ['NOUN', 'ADJ', 'PROPN']:
                        predictions[i - 3 - j] = 'PRD'
                        j += 1
                        if i - 3 - j < 0:
                            break
        if tokens[i + 1] == 'de' and tokens[i + 2] == 'la':
            if pos_tags[i + 3] in ['NOUN', 'ADJ', 'PROPN']:
                predictions[i + 3] = 'PRD'
                j, out = 1, 0
                while pos_tags[i + 3 + j] in ['NOUN', 'ADJ', 'PROPN']:
                    predictions[i + 3 + j] = 'PRD'
                    j += 1
                    if i + 3 + j == len(tokens):
                        out = 1
                        break
                if out == 0:
                    if (tokens[i + 3 + j] == 'es' or tokens[i + 3 + j] == 'son') \
```

```
                        and pos_tags[i + 3 + j + 1] in ['NOUN', 'ADJ', 'PROPN']:
                    predictions[i + 3 + j + 1] = 'PRD'

    if tokens[i + 1] == 'del':
        if pos_tags[i + 2] in ['NOUN', 'ADJ', 'PROPN']:
            predictions[i + 2] = 'PRD'
            j, out = 1, 0
            while pos_tags[i + 2 + j] in ['NOUN', 'ADJ', 'PROPN']:
                predictions[i + 2 + j] = 'PRD'
                j += 1
                if i + 2 + j == len(tokens):
                    out = 1
                    break
            if out == 0:
                if (tokens[i + 2 + j] == 'es' or tokens[i + 2 + j] == 'son') \
                        and pos_tags[i + 2 + j + 1] in ['NOUN', 'ADJ', 'PROPN']:
                    predictions[i + 2 + j + 1] = 'PRD'

# Looking for inductor
elif tokens[i].lower() == 'no':
    if (tokens[i + 1] == 'tiene' or tokens[i + 1] == 'tener')
       and pos_tags[i + 2] == 'NOUN':
        predictions[i] = 'IND'
        predictions[i + 1] = 'IND'
        predictions[i + 2] = 'IND'
    elif (tokens[i + 1] == 'lleva' or tokens[i + 1] == 'llevaba')
         and pos_tags[i + 2] == 'NOUN':
        predictions[i] = 'IND'
        predictions[i + 1] = 'IND'
        predictions[i + 2] = 'IND'
    elif (pos_tags[i + 1] == 'AUX' or pos_tags[i + 1] == 'VERB')
         and pos_tags[i + 2] == 'ADJ':
        predictions[i] = 'IND'
        predictions[i + 1] = 'IND'
        predictions[i + 2] = 'IND'

elif tokens[i].lower() == 'sin':
    if pos_tags[i + 1] == 'NOUN' and tokens[i + 1].lower() not in HARM_RESOURCES \
            and tokens[i + 1].lower() not in PRD_RESOURCES:
        predictions[i] = 'IND'
        predictions[i + 1] = 'IND'
    if pos_tags[i + 1] == 'VERB':
        predictions[i] = 'IND'
        predictions[i + 1] = 'IND'

# Looking for missing Harms
elif tokens[i].lower() in HARM_RESOURCES and pos_tags[i] in ['NOUN', 'PROPN', 'ADJ']:
    # Add that token[i - 1] does not include neither "NO" nor "SIN"
    if tokens[i - 1].lower() != 'no' and tokens[i - 1].lower() != 'sin':
        predictions[i] = 'HARM'
        if pos_tags[i - 1] == 'ADP':
            predictions[i - 1] = 'HARM'

# Looking for HM1
```

```
elif tokens[i].lower() in HM1_RESOURCES and pos_tags[i] not in ['AUX', 'PRON']:
    predictions[i] = 'HM1'
    if pos_tags[i - 1] == 'AUX':
        predictions[i - 1] = 'HM1'
        if pos_tags[i - 2] == 'PRON':
            predictions[i - 2] = 'HM1'
    if pos_tags[i - 1] == 'PRON':
        predictions[i - 1] = 'HM1'

# Looking for HM2
elif tokens[i].lower() in HM2_RESOURCES and pos_tags[i] not in ['AUX', 'PRON']:
    predictions[i] = 'HM2'
    if pos_tags[i - 1] == 'AUX':
        predictions[i - 1] = 'HM2'
        if pos_tags[i - 2] == 'PRON':
            predictions[i - 2] = 'HM2'
    if pos_tags[i - 1] == 'PRON':
        predictions[i - 1] = 'HM2'

# Looking for missing inductors
elif tokens[i].lower() in IND_RESOURCES and pos_tags[i] not in ['AUX', 'PRON']:
    predictions[i] = 'IND'
    if pos_tags[i - 1] == 'AUX':
        predictions[i - 1] = 'IND'
        if pos_tags[i - 2] == 'PRON':
            predictions[i - 2] = 'IND'
    if pos_tags[i - 1] == 'PRON':
        predictions[i - 1] = 'IND'

# Looking for trigger
elif tokens[i].lower() in TRG_RESOURCES and pos_tags[i] not in ['AUX', 'PRON']:
    predictions[i] = 'TRG'
    if pos_tags[i - 1] == 'AUX':
        predictions[i - 1] = 'TRG'
        if pos_tags[i - 2] == 'PRON':
            predictions[i - 2] = 'TRG'
    if pos_tags[i - 1] == 'PRON':
        predictions[i - 1] = 'TRG'

# Looking for missing products
elif tokens[i].lower() in PRD_RESOURCES and pos_tags[i] in ['NOUN', 'PROPN', 'ADJ']:
    predictions[i] = 'PRD'
    if pos_tags[i - 1] == 'ADP':
        predictions[i - 1] = 'PRD'

# Looking for missing products
elif tokens[i].lower() in POP_RESOURCES and pos_tags[i] in ['NOUN', 'PROPN', 'ADJ']:
    predictions[i] = 'POP'
    if pos_tags[i - 1] == 'ADP':
        predictions[i - 1] = 'POP'

# Looking for missing POB
elif tokens[i].lower() in POB_RESOURCES
     and pos_tags[i] in ['NOUN', 'PROPN', 'ADJ', 'NUM']:
```

```
            predictions[i] = 'POB'
            if pos_tags[i - 1] == 'ADP':
                predictions[i - 1] = 'POB'
            if pos_tags[i - 1] == 'DET':
                predictions[i - 1] = 'POB'
                if pos_tags[i - 2] == 'ADP':
                    predictions[i - 2] = 'POB'

    return predictions
```

Appendix C

Hyperparameter Tuning

Head	Dataset	Augment	Epochs	BS	LR	F1
Linear	Original	CWE-F+BT	3	16	5e-5	88,06
Linear	Original	CWE-F+BT	3	16	5e-5	87,8
Linear	Original	CWE-C+BT	3	16	5e-5	87,44
Linear	Original	CWE-C+BT	2	16	5e-5	87,37
Linear	Original	CWE-C	3	16	5e-5	87,27
Linear	Original	OS	3	16	5e-5	87,15
Linear	Original	CWE-F+BT	2	16	5e-5	87,05
Linear	Original	CWE-F+BT	3	16	5e-5	86,9
Linear	Original	CWE-F+BT	3	16	5e-5	86,86
Linear	Original	OS	2	16	5e-5	86,79
Linear	Original	OS	2	16	5e-5	86,79
Linear	Original	CWE-C	3	16	5e-5	86,69
Linear	Original	CWE-C+BT	2	16	5e-5	86,69
Linear	Original	CWE-C-I	2	16	5e-5	86,68
Linear	Original	CWE-F+BT	2	16	5e-5	86,65
Linear	Original	CWE-F	2	16	5e-5	86,64
Linear	Original	CWE-C+BT	2	16	5e-5	86,63
Linear	Original	CWE-C+BT	2	16	5e-5	86,63
Linear	Original	OS	2	16	5e-5	86,54
Linear	Corrected	Back Translation	2	16	5e-5	86,52
Linear	Original	CWE-F	2	16	5e-5	86,49

Table C.1: All the experimentation done on the devel set - I.

Head	Dataset	Augment	Epochs	BS	LR	F1
Linear	Original	CWE-C-I	3	16	5e-5	86,44
Linear	Original	CWE-F	2	16	5e-5	86,38
Linear	Corrected	Back Translation	3	16	5e-5	86,37
Linear	Original	CWE-C+BT	3	16	5e-5	86,24
Linear	Original	CWE-C+BT	3	16	5e-5	86,17
Linear	Original	OS	3	16	5e-5	86,15
Linear	Original	CWE-F-I	3	16	2e-5	86,12
Linear	Original	CWE-F+BT	2	16	5e-5	86,07
Linear	Original	None	3	16	5e-5	86,06
Linear	Corrected	None	4	32	2e-5	85,99
Linear	Original	CWE-F-I	2	16	5e-5	85,89
Linear	Original	CWE-F-I	2	16	5e-5	85,83
Linear	Original	CWE-C	2	16	5e-5	85,83
Linear	Original	CWE-F	3	16	5e-5	85,7
Linear	Original	CWE-F+BT	2	16	5e-5	85,69
Linear	Original	CWE-F+BT	2	16	5e-5	85,68
Linear	Original	None	3	32	5e-5	85,66
Linear	Original	CWE-C-I	2	16	5e-5	85,65
Linear	Original	None	3	16	5e-5	85,59
Linear	Original	CWE-C-I+BT	2	16	5e-5	85,55
Linear	Original	CWE-F-I	4	16	2e-5	85,5
Linear	Original	CWE-C-I	4	32	2e-5	85,45
Linear	Original	CWE-F-I	4	32	2e-5	85,44
Linear	Original	OS-I	2	16	5e-5	85,43
Linear	Original	None	3	16	5e-5	85,4
Linear	Original	CWE-F-I	3	16	5e-5	85,39
Linear	Original	CWE-F-I	3	32	5e-5	85,39
Linear	Original	None	2	16	5e-5	85,39
Linear	Original	CWE-C	2	16	5e-5	85,32
Linear	Original	CWE-C-I	2	16	5e-5	85,26
Linear	Original	CWE-F	3	16	5e-5	85,25
Linear	Original	None	3	32	5e-5	85,25
Linear	Original	CWE-C-I	3	32	5e-5	85,16
Linear	Original	CWE-C-I	4	32	2e-5	85,16

Table C.2: All the experimentation done on the devel set - II.

Head	Dataset	Augment	Epochs	BS	LR	F1
Linear	Original	None	4	32	5e-5	85
Linear	Corrected	None	2	16	5e-5	84,95
Linear	Original	CWE-C-I	2	32	5e-5	84,92
Linear	Original	None	4	32	2e-5	84,81
Linear	Original	CWE-F-I	4	32	2e-5	84,77
Linear	Original	OS-I	2	16	5e-5	84,75
Linear	Original	OS-I	2	16	5e-5	84,75
Linear	Original	CWE-C-I	5	32	2e-5	84,71
Linear	Original	CWE-C-I	3	16	2e-5	84,7
Linear	Original	None	2	16	5e-5	84,59
Linear	Original	None	3	32	5e-5	84,58
Linear	Original	None	3	32	5e-5	84,54
Linear	Original	CWE-F	3	16	5e-5	84,51
Linear	Original	CWE-C-I	4	32	2e-5	84,48
Linear	Original	None	2	16	5e-5	84,46
Linear	Original	None	4	32	2e-5	84,44
Linear	Original	CWE-F-I	4	32	2e-5	84,39
Linear	Original	CWE-F	2	16	5e-5	84,34
Linear	Original	None	4	16	2e-5	84,2
Linear	Original	CWE-C-I	3	16	2e-5	84,19
Linear	Corrected	Back Translation	4	32	2e-5	84,18
Linear	Original	None	3	16	2e-5	84,05
Linear	Original	CWE-C-I	4	32	2e-5	84,01
Linear	Original	None	3	16	2e-5	83,93
Linear	Corrected	None	4	32	2e-5	83,89
Linear	Corrected	None	4	32	2e-5	83,79
Linear	Original	CWE-C-I	2	16	2e-5	83,24
Linear	Original	CWE-F-I	2	32	5e-5	83
Linear	Original	None	4	32	2e-5	82,31
CRF	Original	CWE-F+BT	8	16	5e-5	78,94
CRF	Original	CWE-F	8	16	5e-5	78,51
CRF	Original	CWE-F	7	16	5e-5	77,23
CRF	Original	None	10	16	5e-5	75,86
CRF	Original	None	6	16	5e-5	73,92
CRF	Original	None	15	32	2e-5	69,85

Table C.3: All the experimentation done on the devel set - III.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. 2014. arXiv preprint arXiv:1409.0473.

[2] J. Brownlee and S. Cristina. Machine Learning Mastery. The attention mechanism from scratch. https://machinelearningmastery.com/the-attention-mechanism-from-scratch/, 2021.

[3] José Cañete, Gabriel Chaperon, Rodrigo Fuentes, Jou-Hui Ho, Hojin Kang, and Jorge Pérez. Spanish pre-trained bert model and evaluation data. In *PML4DC at ICLR 2020*, 2020.

[4] Miao Chen, Fang Du, Ganhui Lan, and Victor S Lobanov. Using pre-trained transformer deep learning models to identify named entities and syntactic relations for clinical protocol analysis. In *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering (1)*, 2020.

[5] United States Consumer Product Safety Comission (CPSC). Neiss. https://www.cpsc.gov/Research--Statistics/NEISS-Injury-Data, 2021.

[6] Hospital de Nens de Barcelona. https://hospitaldenens.com/.

[7] J. Devlin, M. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL*, 2019.

[8] J. Devlin, M. Chang, K. Lee, and K. Toutanova. Multilingual bert. https://github.com/google-research/bert/blob/master/multilingual.md, 2019.

[9] DFG, Technische Universitat Darmstadt, and Ubiquitous Knowledge Processing. Inception. https://inception-project.github.io/, 2021.

[10] Hospital Vall d'Hebron. https://www.vallhebron.com/ca.

[11] Doccano. Doccano. https://doccano.herokuapp.com/, 2021.

[12] Alexis Conneau et al. Unsupervised cross-lingual representation learning at scale. 2019.

[13] Y. Liu et al. Roberta: A robustly optimized bert pretraining approach. 2019. arXiv preprint arXiv:1907.11692.

[14] Hugging Face. Transformers. https://huggingface.co/docs/transformers/index, 2021.

[15] Google. Googletrans. https://pypi.org/project/googletrans/, 2020.

[16] Google. Google colaboratory. https://colab.research.google.com/?hl=es, 2021.

[17] Google. Google docs. https://www.google.com/docs/about/, 2021.

[18] Google. Google sheets. https://www.google.com/sheets/about/, 2021.

[19] Fabian Gringel. The best free labeling tools for text annotation in nlp. https://dida.do/blog/the-best-free-labeling-tools-for-text-annotation-in-nlp, 202.

[20] Salvador Merina Herrera. Human language engineering: Advanced machine translation, 2021. Slides from HLE master course from MAI (FIB).

[21] Salvador Merina Herrera. Human language engineering: Information extraction, 2021. Slides from HLE master course from MAI (FIB).

[22] ieConsumo. Programa pipadi. http://www.pipadi.com/index.php, 2021.

[23] ieConsumo S.L. http://www.ieconsumo.org/.

[24] Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. Marian: Fast neural machine translation in C++. In *Proceedings of ACL*

2018, System Demonstrations, pages 116–121, Melbourne, Australia, July 2018. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P18-4020.

[25] T. Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. *ACL*, pages 66–75, 2018.

[26] T. Kudo and J. Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *EMNLP*, 2018.

[27] Ankit Kumar, Piyush Makhija, and Anuj Gupta. Noisy text data: Achilles' heel of bert, 2020.

[28] G. Lample and A. Conneau. Crosslingual language model pretraining. *NeurIPS*, 2019.

[29] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. Neural architectures for named entity recognition. *NAACL*, pages 260–270, 2016.

[30] F. Marzieh, A. Bisazza, and C. Monz. Data augmentation for low-resource neural machine translation. 2017. arXiv preprint arXiv:1705.00440.

[31] MIT. Brat. https://brat.nlplab.org/, 2012.

[32] E.S. Shahul. NeptuneBlog. Data augmentation in nlp: Best practices from a kaggle master. https://neptune.ai/blog/data-augmentation-nlp, 2021.

[33] NVIDIA. Nvidia tesla k80. https://www.nvidia.com/es-es/data-center/tesla-k80/, 2021.

[34] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *NAACL*, 2018.

[35] PSM. Product safety management. https://www.productsafetymanagement.com/es/inicio/, 2021.

[36] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding with unsupervised learning. *Technical report, OpenAI*, 2018.

[37] Ramshaw and Marcus. Text chunking using transformation-based learning. 1995.

[38] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

[39] Stefan Schweter and Alan Akbik. Flert: Document-level features for named entity recognition, 2021.

[40] A. Prakash. Towards Data Science. 3 types of contextualized word embeddings from bert using transfer learning. https://towardsdatascience.com/3-types-of-contextualized-word-embeddings-/from-bert-using-transfer-learning-81fcefe3fe6d, 2021.

[41] Chetna Khanna. Towards Data Science. Wordpiece: Subword-based tokenization algorithm. https://towardsdatascience.com/wordpiece-subword-based-tokenization-algorithm-1fbd14394ed7, 2018.

[42] Karim Raimi. Towards Data Science. Illustrated: Self-attention. https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a#, 2019.

[43] R. Sennrich, B. Haddow, and A. Birch. Improving neural machine translation models with monolingual data. *In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 86–96, 2016.

[44] Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. Bertimbau - portuguese bert. https://github.com/neuralmind-ai/portuguese-bert, 2019.

[45] Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. Portuguese named entity recognition using bert-crf, 2020.

[46] spaCy. Spanish. https://spacy.io/models/es, 2021.

[47] Charles Sutton and Andrew McCallum. An introduction to conditional random fields, 2010.

[48] Erik F. Tjong Kim Sang. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*, 2002. URL https://www.aclweb.org/anthology/W02-2024.

[49] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2003. URL https://www.aclweb.org/anthology/W03-0419.

[50] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez N., Ł. Kaiser, and I. Polosukhin. Attention is all you need,. *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[51] Jason Wei and Kai Zou. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1670. URL https://aclanthology.org/D19-1670.

[52] Wikipedia. Cosine similarity. https://en.wikipedia.org/wiki/Cosine_similarity, 2021.

[53] Wikipedia. Inside–outside–beginning (tagging). https://en.wikipedia.org/wiki/Inside-outside-beginning_(tagging), 2021.

[54] Wikipedia. Long short-term menory. https://en.wikipedia.org/wiki/Long_short-term_memory, 2021.

[55] Wikipedia. Natural language processing. https://en.wikipedia.org/wiki/Natural_language_processing, 2021.

[56] Wikipedia. Named entity. https://en.wikipedia.org/wiki/Named_entity, 2021.

[57] Wikipedia. Wilcoxon signed-rank test. https://en.wikipedia.org/wiki/Wilcoxon_signed-rank_test, 2021.

[58] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame,

Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6.

[59] Y. Wu, M. Schuster, Z. Chen, Q. V. Lee, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, and K. Macherey et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *Technical report, OpenAI*, 2016. arXiv preprint arXiv:1609.08144.

[60] Albert Au Yeung. Bert - tokenization and encoding. https://albertauyeung.github.io/2020/06/19/bert-tokenization.html/, 2020.