Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Thesis in Computer Engineering

Analysis and evaluation of Neural Models for de-identification of privacy-related entities in text documents

Master Candidate

Ghedin Roberto

Student ID 2015135

Supervisor

Prof. Giorgio Satta

University of Padua

Academic Year
2022/2023

Dedicated to my relatives
and friends

Abstract

Digitalization and the high amount of data that companies need to process every day has highlighted the necessity over the years of automated process that can compute long and complex task, without a direct support of people. In this direction, the field of Machine Learning gives an infinite number of combinations of possibility on developing high complex algorithms that can work autonomously with high percentages of reliability.
This work focus the attention on developing a strong architecture able to detect and anonymize specific domains of data (as companies name, person, contact, profession, laws etc..) inside each document. The lack of data available for the italian language created also the necessity of building artificial dataset to perform initial tests on specific domains, but also to explore Transfer Learning technique with multilingual models.
The idea is to build a deep knowledge on the most powerful neural networks (based on state of the art in de-identification) and analyze all the possible pros and cons of each one. The results show in general a good behaviour of all models tested in each experiment. The main part of the analysis are related not only on finding which one has the best accuracy, but also the one with the most convenient balancing between the demand of physical resources (CPU, GPU, weight of the model) with respect to the costs in term of time.

Contents

List of Figures

List of Tables

List of Algorithms

List of Code Snippets

List of Acronyms

NLP Natural Language Processing

Bi-LSTM Bi-directional Long Short Term Memory

CRF Conditional Random Fields

BERT Bidirectional Encoder Representation from Transformers

NER Named Entity Recognition

GloVe Global Vectors

MLM Masked Language Model

NSP Next Sentence Prediction

KIND Kessler Italian Named-Entity Dataset

HREFs Hypertext References

L-BFGS Limited-memory BFGS

LCCRF Linear Chain CRF

1 Introduction

Recent years has seen a remarkable development of automated mechanism, guided by cutting-edge Machine Learning techniques. This gave birth to the great advance of Natural Language Processing (NLP), a branch of artificial intelligence that try to understand, manipulate, generate and process the natural language used by humans. Exploiting powerful architectures such as neural networks inside NLP tasks, it is possible to comprehend human language and to process it over an infinite number of tasks, such as machine translation, Chatbot to mimic human to human conversations, digital assistants that focuses on helping on specific user requests, sequence labelling for text identification of specific entities, etc.. All these features are user-friendly, since they usually operates independently and does not require any prior knowledge of the "black-box" NLP algorithm behind any applications. This advantage makes everything easily applicable to any different field of work where humans can operates. The knowledge of these algorithms is potentially unlimited, since much more context we can transfer to NLP models, the major context they can apply over any proposed problems. The main advantage is surely the time saving, where manual task done by hand can be automated and finished much more faster. For instance, a text translation from a source to a target language can require hours or even days, depending on the number of pages of the text. A neural networks trained on machine translation can solve the same operation in few minutes. This underline why companies started investing on NLP, since the benefits in term of production can be incredibly high, starting from the speeding up of business processes, to the high savings of resources (number of personnel, money etc..).

Anyway, building optimal models performing NLP tasks is not that easy. The human language is extremely complex and it is constantly evolving. We just need to think at how many different languages (and related dialects) exist and how many ways of interpreting each phrase there could exist. For this reason, NLP models usually require high amount of data to understand and to react to human needs.
One of the most recent and famous result of NLP is ChatGPT, an extremely intelligent ChatBot that is able to simulate almost to perfection a conversation between humans. It is trained to answer basically to every possible question that someone can think. ChatGPT is so powerful that many universities banned it to avoid that students use it as a tool for thesis or other projects. This also opens to a big world of ethical problems linked to NLP models (such as privacy data preservation, racism contents, etc..). The high speed of evolution of technology makes this situation one of the biggest problem of the recent years, that will surely lead to more and more considerations over times, trying to maintain a good balance between human rights and the knowledge power of artificial intelligences.

2 Theoretical Background

2.1 Sequence Labelling and NER

Named Entity Recognition (NER) is a subtask of a wider problem called *Sequence Labelling*. The aim is to assign to a given input word sequence $a_1, a_2, \ldots, a_n$ an output sequence $b_1, b_2, \ldots, b_n$ of arbitrary pre-defined categories based on context. NER seeks to locate specific span of words referring to entities such as person, organization, addresses, time expressions etc..

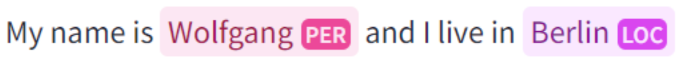

My name is Wolfgang PER and I live in Berlin LOC

Figure 2.1: NER example

The standard specific for span-recognition is called BIO tagging:

- all first tokens of all the entities are marked as B
- tokens that occur inside entities are marked as I
- tokens outside entities are marked as O

```
EU B-ORG
rejects O
German B-MISC
call O
to O
boycott O
British B-MISC
lamb O
. O
```

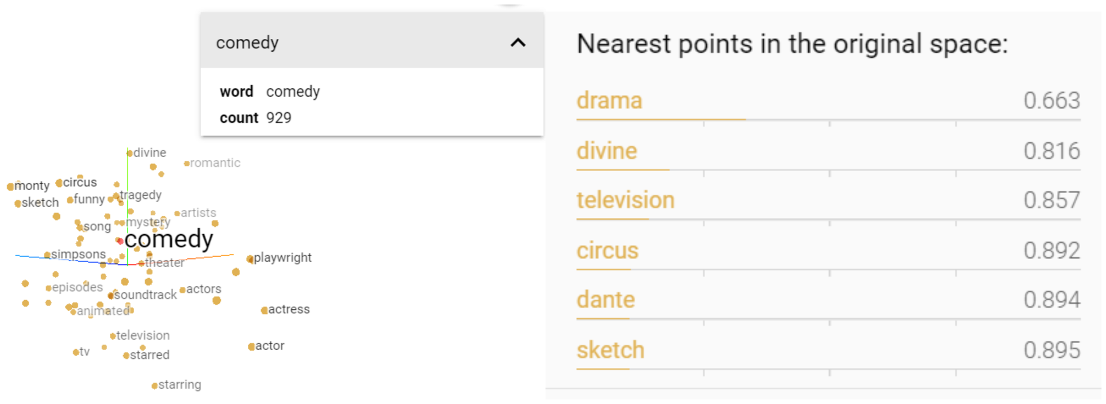
Figure 2.2: BIO tagging example

Many other variation from BIO tagging has been defined, such as IO and BIOES, where the way to represent entities slightly change. IO tagging simplifies the tagging assumption without specifying directly the starting token of an entity (every token inside a span is marked as I and O if outside). Instead, BIOES tagging is used to strongly highlights all parts of entities span:

- all first tokens of all entities are marked as B
- tokens inside a span, except for the first and last one, are marked as I
- tokens that end a span are marked as E
- entities defined by a single word span are marked as S

2.2 Word Embeddings

Word embeddings is a term related to the representation of words through vectors of real values.



Nearest points in the original space:

drama	0.663
divine	0.816
television	0.857
circus	0.892
dante	0.894
sketch	0.895

Figure 2.3: Word embeddings projector

These values encode the meaning of each word and the principle is that words with similar meanings should be close inside the vector space. In Fig. 2.3 is shown an example of a 3D representation the vector space, referring to the word "Comedy".
There are many techniques to produce word embeddings, such as probabilistic methods, term-context matrix, matrix approximation on the co-occurrence matrix. Furthermore, word embeddings can be produced by neural networks. Their characteristics are:

- smaller number of vectors dimension (typically 100, 200 or 300)
- components can be also negative
- there is not a direct rational connection between dimension of vectors and the context of words

The way neural networks learn semantic and context of words is much more complex and implicitly they are able to extract further semantic informations for each word, creating a more calibrated vector space representation.
It is possible to divide these types of embeddings in two classes:

- static embeddings - each word in the dictionary is represented by one fixed vector.
- contextual embeddings - every word occurrence is represented by different vectors, depending on the context.

In next sections, all the embeddings used specifically for this project are introduced.

2.2.1 FastText

FastText[1] is an embedding based on the skipgram model, where each word is represented by itself plus a bag of words of character N-grams[2]. The final embedding will be the sum of all these representations.
Skipgram algorithm (Mikolov et al., 2013 [9]) generates embeddings based on a target word and context words. Given a vocabulary of size W, the goal is to

[1] `https://fasttext.cc/docs/en/crawl-vectors.html`
[2] a N-sequence of characters of a word

compute a vectorial representation for each word $w \in W$. Formally, given a large corpus represented by a sequence of words $w_1, ..., w_t$, the objective function to be maximized for the skipgram model is

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{-c<=j<=c, j \neq 0} \log p(w_{t+j}|w_t) \tag{2.1}$$

where c is the size of training context. Basic Skipgram algorithm defines the probability $p(w_{t+j}|w_t)$ using softmax function:

$$p(w_O, w_I) = \frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^{\top} v_{w_I}\right)} \tag{2.2}$$

where v_w and v'_w are the input and output (these are also called target and context embedding) vector representation of w and W is the size of the vocabulary. The computation results impractical, since it is directly proportional to $|W|$, which is often a very high value. To fix this problem, Negative Sampling has been proposed, which approximates the probability of the softmax. The idea is to to use logistic regression to train the model to distinguish between positive and negative examples. The Negative Sampling objective becomes

$$\log \sigma({v'_{w_O}}^{\top} v_{w_I}) + \sum_{i=1}^{k} {}_{w_i \sim P_n(w)} [\log \sigma(-{v'_{w_i}}^{\top} v_{w_I})] \tag{2.3}$$

where $P_n(w)$ is the noise distribution, containing k negative examples for each data sample. Experiments suggest as optimal values for k the range from 2 to 20, based also on the size of training datasets. The optimization of the model is solved using stochastic gradient descent, maximizing for each step the similarity between target and context embeddings and decreasing the correlation between words and noise examples.

2.2.2 Global Vectors (GloVe)

Global Vectors (GloVe) is an embedding that uses a weighted least square model trained on word to word co-occurences, leading to an high efficient use of statistic informations of the corpus. The aim is to exploit the idea of methods based on term-context matrix (e.g. PPMI (Positive Pointwise Mutual Information) or LSA (Latent Semantic Analysis) and to mix it with methods like the skipgram model. This is due mainly to the significant drawbacks of both the two types of model used singularly. PPMI or LSA high concentrate on statistical information without taking into account the word analogy task. Conversely, skipgram model or similar works pretty good on the analogy task, but poorly concentrate on corpus statistics. Results on the original paper [12] demonstrates the effectiveness of this embedding, outperforming previous methods in many tasks, including NER.

2.3 Models

2.3.1 Conditional Random Field (CRF)

Conditional Random Fields (CRF) is a discriminative sequence model to segment and label data. It is able to distinguish different classes, but it is not able to generate examples. The most used version for language processing is called Linear Chain CRF (LCCRF). Formally, given the input word sequence $x_1, x_2, ..., x_n = x_{1:n}$, and given $Y(x_{1:n})$ be the set of all the possible tag sequences $y_1, y_2, ..., y_n$, LCCRF solves the following problem:

$$\hat{y}_{1:n} = arg\,max_{y_{1:n} \in Y(x_{1:n}} P(y_{1:n}|x_{1:n}) \tag{2.4}$$

At each step, the models computes log-linear functions over a set of local features that are subsequently aggregated to produce the final global probability. Defining K global features functions $F_K(x_{1:n}, y_{1:n})$ with their relative weights w_k, we get

$$P(y_{1:n}|x_{1:n}) = \frac{\exp\left(\sum_{k=1}^{K} w_k F_k((x_{1:n}, y_{1:n})\right)}{\sum_{y'_{1:n} \in Y(x_{1:n})} \exp\left(\sum_{k=1}^{K} w_k F_k(x_{1:n}, y'_{1:n})\right)} \tag{2.5}$$

where the denominator, called partition function, is a normalization factor.

$$Z(x_{1:n}) = \sum_{y'_{1:n} \in Y(x_{1:n})} \exp\left(\sum_{k=1}^{K} w_k F_k(x_{1:n}, y'_{1:n})\right) \tag{2.6}$$

For CRF training, negative log-likelihood with L1 or L2 regularization is used as objective function. Given the train dataset $D = \{(y^{(h)}_{1:n_h}, x^{(h)}_{1:n_h})|1 <= h <= N\}$, where $x^{(h)}_{1:n_h}$ are the sentences and $y^{(h)}_{1:n_h}$ are the associated sequence label, the objective function is

$$L(D,w) = \frac{\lambda}{2}||w^2|| - \sum_{h=1}^{N} \log P(y^{(h)}_{1:n_h}, x^{(h)}_{1:n_h}) \tag{2.7}$$

and can be optimized using stochastic gradient descent. Finally, decoding phase can be efficiently solved exploiting Viterbi algorithm, a dynamic programming framework that, given a sequence of observation, computes the most likely path (sequence) of tags.

2.3.2 Bi-Directional LSTM + CRF

Long Short Term Memory (LSTM) are a derivation of Recurrent Neural Networks (RNN). RNN are trained to learn time varying patterns. They are composed by feedforward closed loop connection, making them able to capture long-distance dependencies. In practice, sometimes they fail due to the exploding/vanishing problem [1]. The *exploding* gradient problem is related to a large increase in the norm of the gradient during training. These events are caused by the explosion of long term components, that can easily grow exponentially. The *vanishing* gradient problem is the inverse behaviour, where the long term components norm goes exponentially fast to 0, making impossible for the model to learn temporally distant events. This problem has been solved with the introduction of LSTM, an evolution of RNN with an appropriate gradient-based algorithm.

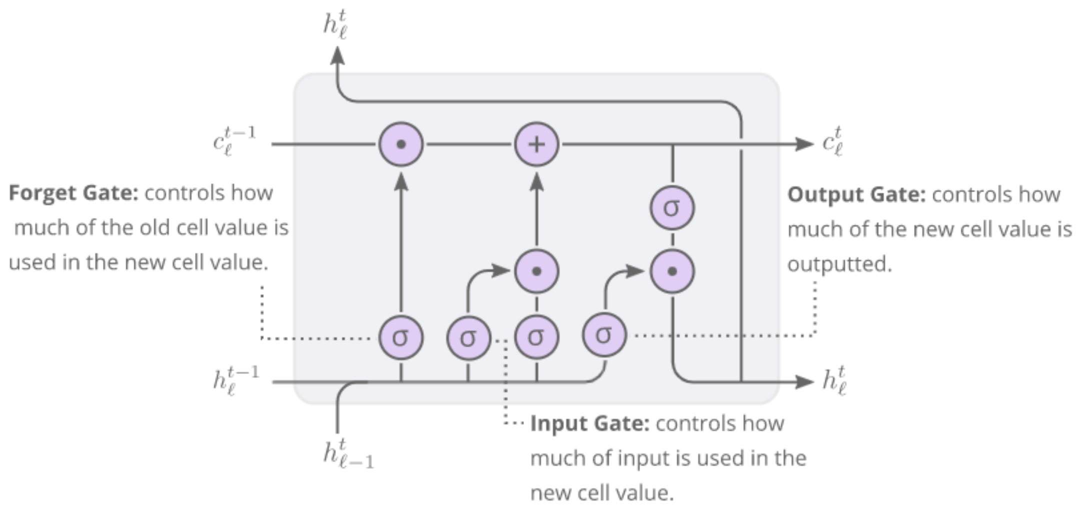

h_ℓ^t
c_ℓ^{t-1}
c_ℓ^t
Forget Gate: controls how much of the old cell value is used in the new cell value.
Output Gate: controls how much of the new cell value is outputted.
h_ℓ^{t-1}
h_ℓ^t
$h_{\ell-1}^t$
Input Gate: controls how much of input is used in the new cell value.
σ
+

Figure 2.4: LSTM Architecture

In Fig 2.4 we can see a general example of a LSTM unit. The main difference between RNN and LSTM is the presence of three multiplicative gates, controlled by a sigmoid function that acts as a 0/1 switch to activate or deactivate them. The gates are:

- forget gate - it selects how much is important the value contained in the previous cell, deciding the portion of previous information to forget/retain
- input gate - it allows the input information to flow in the memory cell
- output gate - control the portion of the current value to be read in output

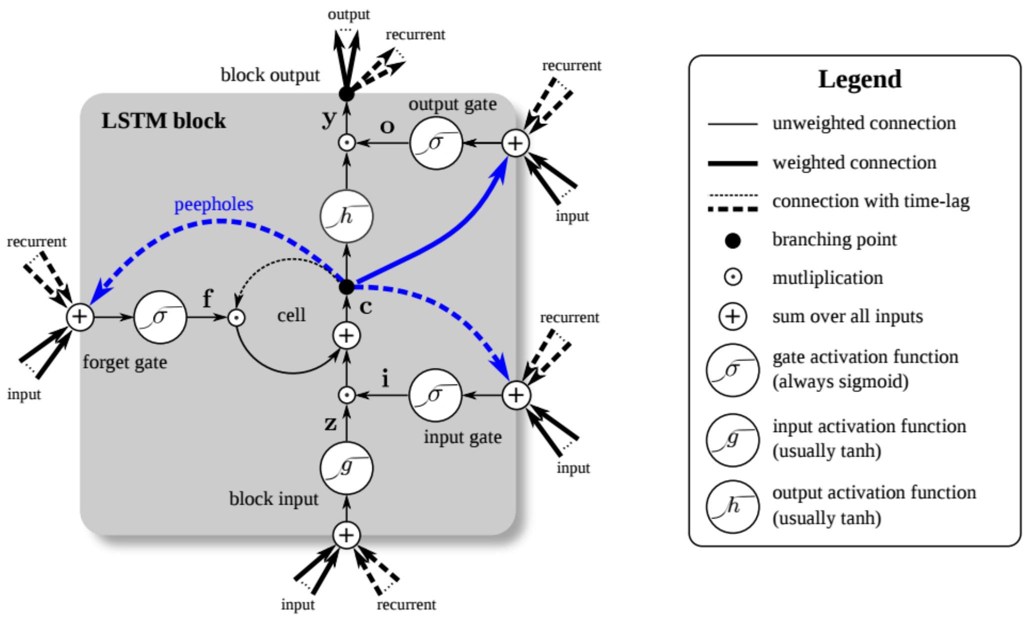


Figure 2.5: LSTM unit with peephole connections

The LSTM architecture can be also improved by adding peephole connections (Fig 2.5), that lead to a control of all gates to allow for easier learming of precise timings. Formally, the general passes to update a *vanilla* LSTM unit are (at time t):

$$\begin{aligned}
i_t &= \sigma(W_i h_{t-1} + U_i x_t + b_t) \\
f_t &= \sigma(W_f h_{t-1} + U_f x_t + b_f) \\
\hat{c}_t &= \tanh(W_c ht - 1 + U_c x_t + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \hat{c}_t \\
o_t &= \sigma(W_o h_{t-1} + U_o x_t + b_o) \\
h_t &= o_t \odot \tanh c_t
\end{aligned}$$

where:

- σ is the sigmoid function
- $\odot$ is the element-wise product
- x_t and h_t are respectively the input vector and the hidden state at time t
- W_i, W_f, W_c, W_o are the weight matrices for the hidden state h_t
- U_i, U_f, U_c, U_o are the weight matrices for each gates for input x_t
- b_i, b_f, b_c, b_o are the bias vectors

The characteristics LSTM are really suitable for tasks as NER, speech recognition, etc. An elegant solution to empower this model is to use a double layer of LSTM, one looking at the sequence forwards and the other one backwards. This model is called Bi-directional LSTM and is able to capture the past and future information inside data. The two layers will produce two different hidden states that are finally concatenated. The input received by the model are the word embeddings (in this case static embeddings) plus the character-level embedding of each word.

Previous works (Santos and Zadrozny, 2014 [5]; Chiu and Nichols, 2015 [2]) showed the effectiveness of extracting character representation of words, leading to a better morphological comprehension. This experiments can be efficiently made exploiting both Convolutional Neural Networks (CNN) or a Bi-LSTM (as in Fig. 2.6). The character-level representation is then concatenated with pre-trained word embeddings for each distinct word of the vocabulary and used as input to feed the model.

The importance of pre-trained word embeddings has been underlined by previous works (Collobert et al., 2011 [3]; Ma et al., 2016 [8]). They made a comparison between randomly initialized word vectors against some well known publicy available pre-trained word embeddings (as Glove, FastText etc..). As expected, results show the high improvement of performances with respect to the random word embeddings, especially on NER tasks. Given in input to the model the contextual representation of words, the final output will be the score for each possible entities (for each token), where the higher value indicates the best solution found by the model. These score values are called *emission probabilities*. Even if these probabilities could be enough to produce a final prediction on the sequence of tags (just by taking the higher probability for each token and assign to it the correspondent entity), usually a final CRF is added.

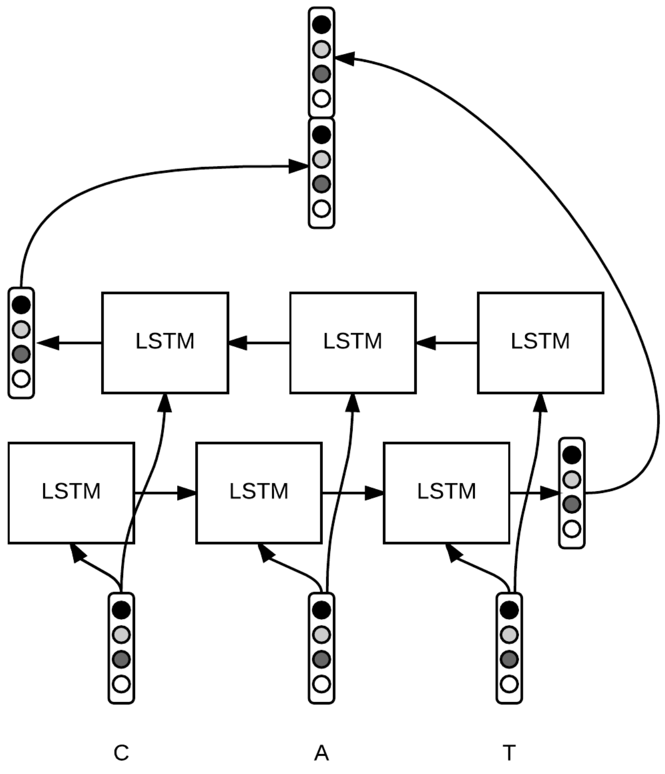


Figure 2.6: Character-level representation

This is due mainly to the inability of Bi-LSTM to completely understand the context and all the inter-correspondences between "similar" but very different words (e.g. "New York" is a complete different entity with respect to "New York Times"). This problem is highly solved by the CRF layer, able to understand the main transitions between entities and especially to reproduce the correct BIO tagging transitions (e.g. an O label cannot be followed by an I label). The decoding is made through the Viterbi algorithm, that calculates and predicts the best sequence path of the tags.

2.3.3 BERT + CRF

Bidirectional Encoder Representation from Transformers (BERT) is a revolutionary language representation model, designed to predict words by jointly conditioning left and right context. (Devlin et al., 2018 [4]), producing for each

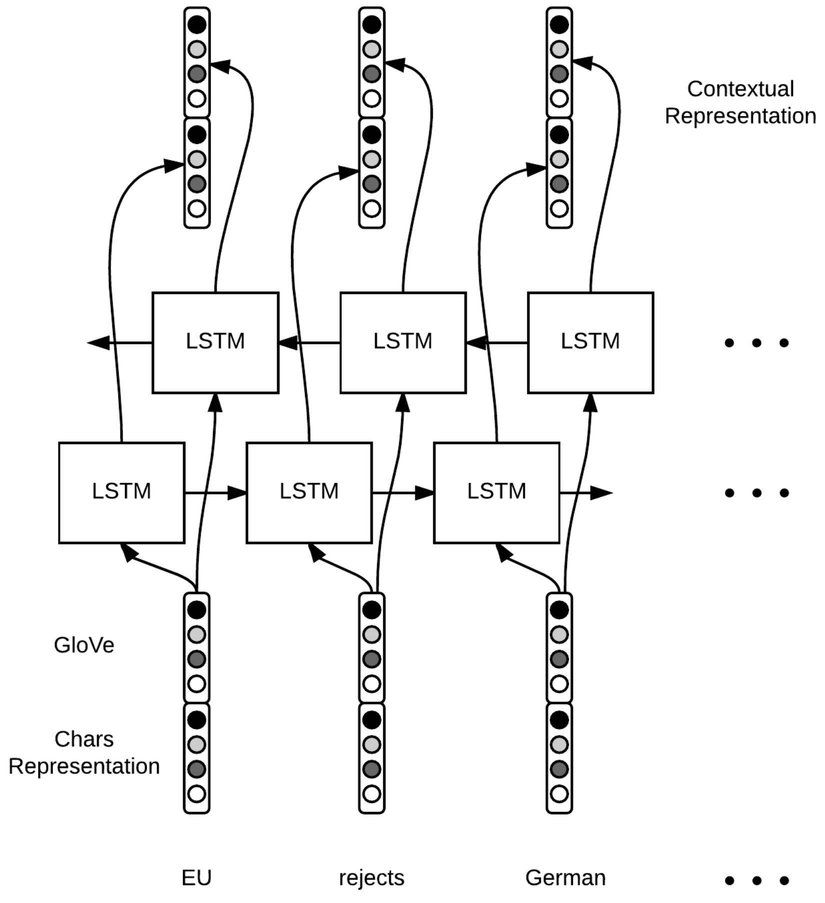


Figure 2.7: Concatenation of character-level and GloVe word embeddings

token its vector representation (dynamic embedding). Other language models has been created, but they showed evident limitations with respect to BERT. The main reason is due to the general unidirectional architecture (only left to right). This one can be optimal for sentence-level tasks, but experiments highlights the loss of performances in tasks such as NER or question answering, where is fundamental to extract both left and right context informations.
BERT architecture is a multi-layer bidirectional Transformer encoder, based on the structure of the original paper related to Transformers neural networks (Vaswani et al., 2017 [14]).
The input/output are segmented through sub-word tokenization[3] and com-

[3]Process that split phrases/sentences into smaller units, called tokens.

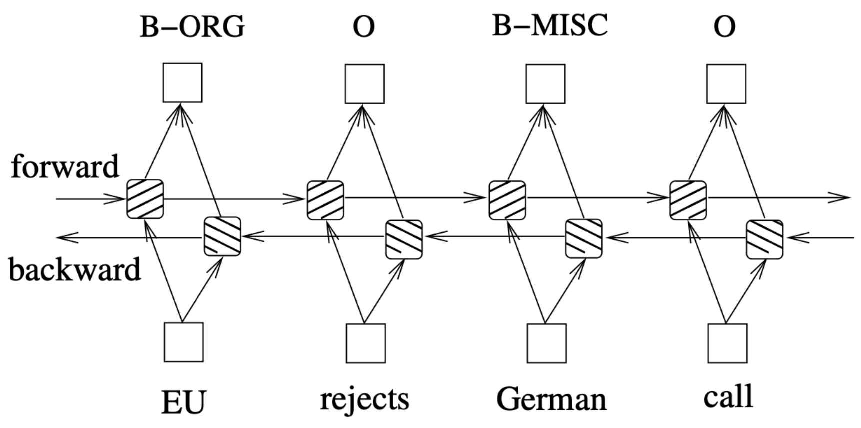


Figure 2.8: Bi-LSTM + CRF model

bined with positional embeddings and the WordPiece token embeddings (Wu et al., 2016 [15]), as showed in Fig. 2.9.

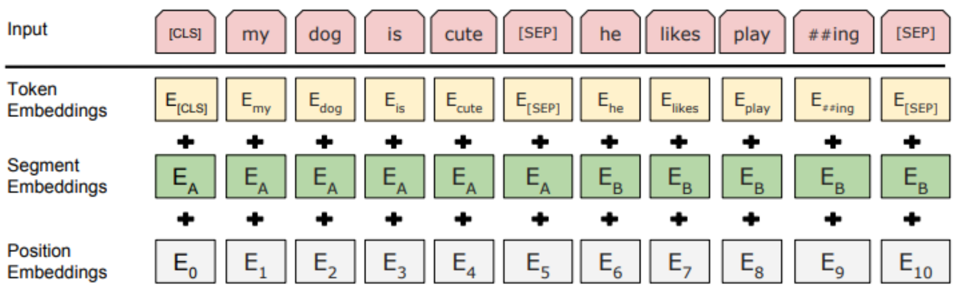
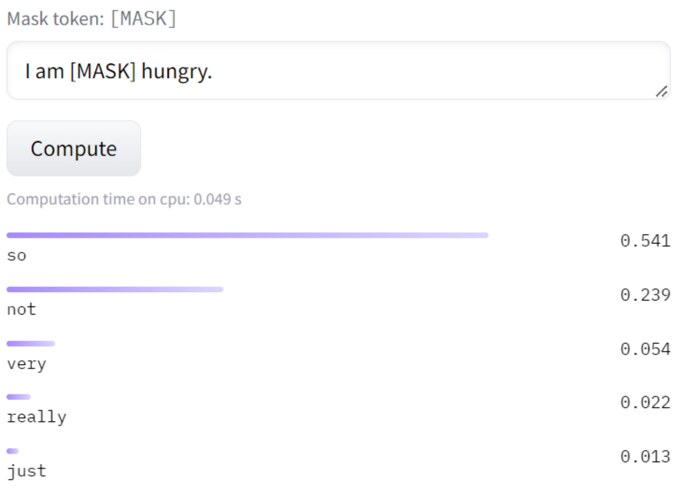

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 2.9: BERT input representation

The pre-training of the model is not made through traditional left-to-right language models, but solving two unsupervised tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP).
With MLM, BERT learn to perform a "fill in the blank" prediction (Fig 2.10, where the 15% of tokens chosen at random are:

- replaced with the special token [MASK] (80% of times)
- replaced with another random token (10% of times)
- left unchanged

The main reason of this procedure is to mitigate the effectiveness of the [MASK] token, since it is used only in pre-training and not in the fine-tuning phase.



Mask token: [MASK]

I am [MASK] hungry.

Compute

Computation time on cpu: 0.049 s

```
so        0.541
not       0.239
very      0.054
really    0.022
just      0.013
```

Figure 2.10: Masked Language Modeling example

NSP, instead, is really beneficial to train the model to work with downstream tasks (such as Question Answering or Natural Language Inference), that are based on understanding the relationship between two phrases. Given a generic pair of phrases, the first token is always a special token [CLS] and they are separated by another unique token [SEP]. Each pair of them are 50% of times an adjacent combination, while in the remaining 50% of cases the two phrases are randomly sampled.

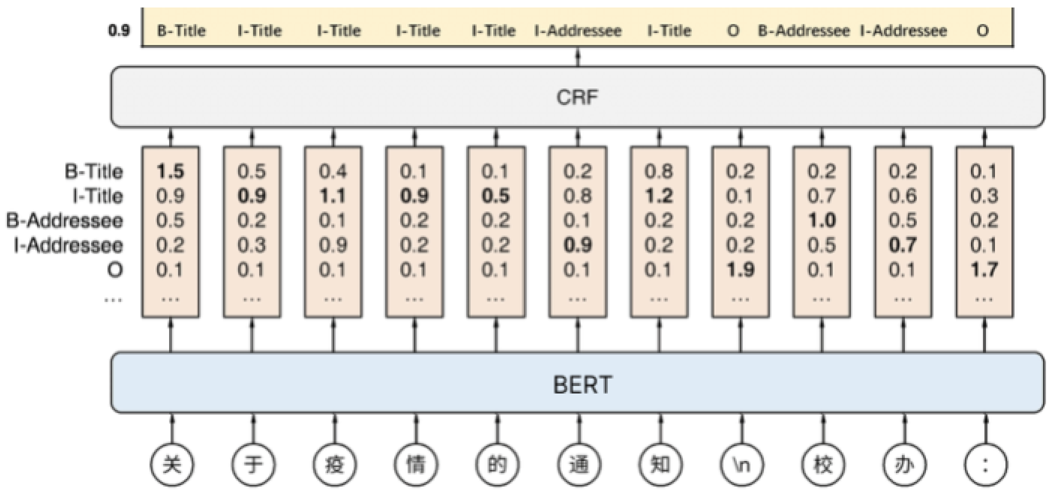


Figure 2.11: Computation of probabilities for BERT-CRF model

In the case of NER, as for the Bi-LSTM model, a final CRF layer is used to boost the performances. BERT produces the emission probabilities that are passed and decoded by the CRF through the Viterbi algorithm, which computes the most likely tags sequence over the entire sentences (Fig. 2.11). As we will see in chapter 4, the CRF layer does not provide a great improvement on the accuracy of the model with respect to other experiments lead with the Bi-LSTM-CRF. This is due to the major power, complexity (higher number of parameters to be trained) and flexibility of BERT, able to capture and analyze better the context inside sentences.

3 Task and Dataset

3.1 De-Identification

This project will exploit the power of neural networks, in relations with NLP techniques, to face up to the De-identification of sensitive data inside documents. De-identification is pretty similar to another well known NLP task, called NER. The main difference is that NER only seeks to locate named entities inside unstructured text, while de-identification also considers the level of anonymization on data. Data can be anonymized completely, so there is no possibility to retrieve any type of information (anonymization) or they can be partially deleted (for example, companies name can be replaced by the word Organization). In this case, entities can be retrieved by the context inside the document.
The idea is to highlight the strengths and weaknesses of both models and understand what is the most adequate to the situation.
The lack of Italian gold dataset for some types of privacy data (like laws and politic parties) has introduced the necessity to create artificial silver Italian dataset, required mainly to provide some kind of test set. Since training set and validation set remain hypothetically in languages different from the Italian, the attention has been placed on Transfer Learning techniques and multilingual models support.
The motivations of this work are rooted to the necessity for companies to preprocess documents and automatically detect all the various domains requested, trying to limit as much as possible the manual work made by human.

Therefore, the final goal starting from this project is to develop a software that automatically detects in each document all the possible type of domains inside each of them.
The project follows a logic sequence of experiments that has been defined with a critic selection of the most important research papers that could be exploited to understand the state of the art related to De-identification. The most interesting studies lead to the application of two types of architectures: Bi-directional Long Short Term Memory (Bi-LSTM) with a final layer of CRF and BERT again with final CRF layer. The final part cover entirely the study of the application of multilingual models from a source language (in this case german and english) to the target language (italian).
Results show the major power and flexibility of BERT models. They are clearly much more complex neural networks and they can be trained to work pretty well also in multilingual cases, with a loss estimated around the 10% w.r.t. performances of monolingual tests. The cons are related mainly to a high demand of resources to receive predictions on data in reasonable time.

3.2 Dataset

This chapter will treat about all the dataset used for this project. The idea is to analyze and discover the most used and accurate dataset inside the most famous papers related to NER or de-identification tasks. We will discuss also about the difficulty on finding accurate dataset for the Italian language. Indeed, generically papers refers always to English dataset with the same categories of entities (person, location, organization, etc.), making it difficult to retrieve useful information to do some comparison on different ones that are required for our tests. Because of this problems, some dataset has been created from scratch, extracting useful information on the internet used to provide a satisfying number of sentences for train and test set.
First experiments are related mainly on gold dataset on the classic entities of NER to ensure a good behavior of the implemented models, while following experiments will explore also Italian dataset, both on classic and unusual entities.

3.2.1 CONLL2003 Dataset

CONLL2003 Dataset (Sang et al., 2003 [13]) is the most used dataset in the field of NER. It is available in two different languages: English and German. For each of the languages all data has been split in train, validation and test set.
The English data was taken from the Reuters Corpus[1] and it contains Reuters news that goes from August 1996 to August 1997.

English data	Articles	Sentences	Tokens
Training Set	946	14,987	203,621
Development set	216	3,466	51,362
Test set	231	3,684	46,435

English data	LOC	MISC	ORG	PER
Training Set	7140	3438	6321	6600
Development set	1837	922	1341	1842
Test set	1668	702	1661	1617

Table 3.1: Statistics of English dataset

German data was taken from the ECI Multilingual Text Corpus[2]. It consists of texts in many languages. The portion of data selected was taken from the German newspaper Frankfurter Rundshau. In Table 3.1 and 3.2 are listed all main statistics of train, validation and test set for both languages.
The dataset structure is the classic format composed by one token per line. Empty lines indicates sentences boundaries. Each line contains four fields: the word, its part-of-speech tags, its chunk tags and its entity tag. Relatively to this project, we will be interested only in the word and its entity tag, discarding other columns. The order follows the BIO tagging rules and five types of entities are defined: persons (PER), organizations (ORG), locations (LOC), miscellaneous (MISC) and O (outside) tag as usual to indicates no entity words.

[1]`http://www.reuters.com/researchandstandards/`
[2]`http://www.ldc.upenn.edu/`

English data	Articles	Sentences	Tokens
Training Set	553	12,705	206,931
Development set	201	3,068	51,444
Test set	155	3,160	51,943

English data	LOC	MISC	ORG	PER
Training Set	4363	2288	2427	2773
Development set	1181	1010	1241	1401
Test set	1035	670	773	1195

Table 3.2: Statistics of German dataset

3.2.2 Jobstack Dataset

	Train	Dev	Test	Total
# Documents	313	41	41	395
# Sentences	18,055	2082	2092	22,219
# Tokens	195,425	22,049	21,579	239,053
# Entities	4,057	462	426	5,154
avg. # sentences	57.68	50.78	51.02	53.16
avg. tokens / sent.	10.82	10.59	10.32	10.78
avg. entities / sent.	0.22	0.22	0.20	0.21
density	14.73	14.31	14.58	14.54
Organization	1803	215	208	2226
Location	1511	157	142	1810
Profession	558	63	64	685
Contact	99	10	7	116
Name	86	17	5	108

Table 3.3: Statistics of Jobstack Dataset

Jobstack is a corpus related to de-identification of personal data in job vacancies (Jensen et al., 2021 [7]). It derives from 395 documents selected over 2,775 job posting from StackOverflow. Sentences have been manually annotated with the classic BIO tagging, splitting tokens in six different entities:

- **Organization** - This includes all citations of companies (related or not to job posting)
- **Location** - It indicates address of company, but also refers to zip codes, cities, regions and countries
- **Contact** - This entity contains URLs, email addresses and phone numbers (e.g. contact info of an employee)
- **Name** - It includes all names of people inside text, no matter the relationship to job postings. Titles such as "Dr." or "Mr." are kept outside this entity.
- **Profession** - It covers all possible professions inside the text. Mentions of colleague position are not listed inside this entity. In case of citations of sequential multiple positions (e.g. "Software/Security Engineer"), the complete string will be annotated as a single profession.
- **O** (Outside) - It indicates all tokens outside entities.

Additionally, a qualitative annotation has been developed analyzing a sample of the data annotated by three different annotators. The agreement is computed over 1500 overlapping sentences using Cohen's κ (Fleiss and Cohen, 1973 [6]) between pairs of annotators and Fleiss' κ (Fleiss, 1971 [6]) to generalize over all annotators. Table 3.4 shows three levels of computations: **Token**, **Entity** and **Unlabeled**. **Token** level look at the agreement of annotators on each token (O tag included). **Entity** refers to the agreement between named entities alone. **Unlabeled** computes the agreement on the exact span match without considering the type of entity. Coefficient values obtained in the analysis confirm a strong agreement between annotators.

	Token	Entity	Unlabeled
A1-A2	0.889	0.767	0.892
A1-A3	0.898	0.782	0.904
A2-A3	0.917	0.823	0.920
Fleiss κ	0.902	0.800	0.906

Table 3.4: Cohen's and Fleiss' κ agreement

3.2.3 KIND Dataset

Dataset	PER	ORG	LOC	Total
Wikinews	8,928	7,593	6,862	247,528
Fiction	3,439	182	733	170,942
Aldo Moro	1,459	4,842	2,024	309,798
Alcide De Gasperi	1,129	2,396	1,046	117,997
Total	14,955	15,013	10,665	846,265

Table 3.5: KIND train set statistics

Dataset	PER	ORG	LOC	Total
Wikinews	1,802	1,823	1,711	61,094
Fiction	636	284	463	21,506
Aldo Moro	282	934	807	82,806
Alcide De Gasperi	253	533	274	32,635
Total	2,973	3,574	3,255	198,041

Table 3.6: KIND test set statistics

Kessler Italian Named-Entity Dataset (KIND)[3] is an Italian dataset for NER (Paccosi and Palmero, 2021 [10]). It contains more than one million tokens, where 600K of them are manually annotated[4], following the classic BIO tagging scheme. It is divided in four different classes of entities:

- **Person** (PER) - It refers to proper names related to individual human being, animal, fictitious person or proper names which refer to a group of person belonging to the same family. Titles or apposition are not considered as part of this entity.
- **Organization** (ORG) - It cover every type of established associations. These ones can be governmental, educational, religious, related to sports, medical-scientific etc.
- **Location** (LOC) - It includes every example related to geographical places or entities which possess physical location and a proper name, such as nations, cities, continents, facilities, bar, restaurants etc.

[3]`https://github.com/dhfbk/KIND`

[4]Part of the text were already annotated and a semi-automatic process has been used just to check the coherence of the classes defined for KIND

The construction of the dataset is based on four chapters with text taken from: Wikinews[5], Fiction books, writings and speeches from Aldo Moro and Alcide de Gasperi. Tables 3.5 and 3.6 gives all the statistical information of train and test set.

Inter-annotators agreement is measured using the Cohen's κ(Fleiss and Cohen, 1973 [6]) between two expert linguist that annotates the same set of 15 documents randomly selected from Wikinews. Results show $\kappa = 0.952$, indicating a strong agreement.

Note Since the complete corpus has a large number of tokens and sentences, for this project we decided to reduce its dimension and create a lighter version of it, to balance the use of resources. The training set is a combination of the training sets related to De Gasperi and Moro, validation is composed by the test set of Moro's speeches and the test set contains De Gasperi's test set.

3.2.4 LEGAL-NER Dataset

Legal-NER[6] is a dataset of German legal documents for NER. It is composed by court decision from 2017 and 2018, published by the Federal Ministry of Justice and Consumer Protection. The dataset consists of 66,723 sentences with over two millions of tokens, selected over seven different federal courts. It includes two different types of annotations (both following BIO tagging scheme), one with 19 fine-grained classes and the other one with 7 coarse-grained classes (Table 3.7)

3.2.5 REDIT Dataset

REDIT dataset (Paccosi and Palmero, 2021 [11]) is a selection of documents taken from different institutions of the public administration. These documents are related to different types of forms, such as licence for parking, school enrollments, marriage licences and so on. It is divided in two parts: one with documents including real data and another one with documents compiled with

[5]It is a multi-language free-content project of journalism. The Italian part is composed by 11K news articles. For KIND, only 1000 of them has been selected randomly to be part of the corpus

[6]`https://github.com/elenanereiss/Legal-Entity-Recognition`

Classes	#	%
Person (PER)	1,747	3.26
Judge (RR)	1,519	2.83
Lawyer (AN)	111	0.21
Person (PER)	**3,377**	**6.30**
Country (LD)	1,429	2.66
City (ST)	705	1.31
Street (STR)	136	0.25
Landscape (LDS)	198	0.37
Location (LOC)	**2,468**	**4.60**
Organization (ORG)	1,166	2.17
Company (UN)	1,058	1.97
Institution (INN)	2,196	4.09
Court (GRT)	3,212	5.99
Brand (MRK)	283	0.53
Organization (ORG	**7,915**	**14.76**
Law (GS)	18,520	34.53
Ordinance (VO)	797	1.49
EU Legal Norm (EUN)	1,499	2.79
Legal Norm (NRM)	**20,816**	**38.81**
Regulation (VS)	607	1.13
Contract (VT)	2,863	5.34
Case-by-c. regul.	**3,470**	**6.47**
Court decision (RS)	**12,580**	**23.46**
Legal literature (LIT)	**4,006**	**5.60**
Total	53,632	100

Table 3.7: Coarse and fine-grained entities of Legal NER dataset

fictitious data, aiming to avoid using sensitive data in terms of privacy issues (this make the dataset publicly available). Table 3.8 show all the different entities (with their relative number) defined inside the two dataset (the real and the fictitious one),

3.2.6 Artificial LEX Dataset

As already mentioned, Italian language is not really rich of data already annotated related to entities that goes outside from the classical ones (Person, Organization, Location). Speaking about the public administration, it is possible to find conspicuous amount of available (not annotated) free text. Due to the project organization and the restricted times to select some linguists that

Entities	Fict. dataset	Real dataset
ENTE	192	2829
LEX	214	8314
LOC	743	3788
ORG	62	2179
PER	228	4197
Total	1,439	21,307

Table 3.8: Amount of entities annotations inside REDIT dataset

annotates the data, an automatic process has been developed to create a silver dataset related to references of laws. Research work lead to an interesting site of the "Ministero dell'Economia e delle Finanze"[7] (MEF). It has a judgements archive that goes from 2015 to 2022[8]. The interesting thing is that laws inside texts are linked directly to the documentation of the legislation. The following text shows a brief example.

> *2.- Con il secondo motivo del ricorso principale si lamenta ai sensi dell'art. 360, n. 3, la violazione del D.Lgs. n. 504 del 1992, art. 7 lett. i, ribadendo la sussistenza dei requisiti oggettivi e soggettivi per fruire dell'agevolazione. La parte deduce che la Commissione regionale ha omesso di verificare in termini rigorosi le caratteristiche dell'attività cui l'immobile è destinato, facendo riferimento ad un accertamento "in via del tutto induttiva ma comunque plausibile" al fine di escluderne la natura non commerciale.*

The idea is to exploit URLs links and extract LEX entities to create automatically an annotated dataset, following as usual the BIO tagging scheme. All the process requires primarily the extraction (for each document) of all the HTML texts from the site. This is done through the implementation of a scraper that given the URL link of the archive judgement, it downloads all the available documents of that specific year.

```python
from requests_html import HTMLSession,AsyncHTMLSession
from bs4 import BeautifulSoup
import requests
from tqdm import tqdm
```

[7]https://www.finanze.gov.it/it/

[8]https://www.giustiziatributaria.gov.it/gt/web/guest/rassegna-sentenze-tributarie

```python


def render_JS(URL,session):
    r = session.get(URL)
    r.html.render()
    return r.html.html

#retrieve the URL to access the desired text
def get_complete_sentence_link(url):
    initial_page = requests.get(url)
    content = initial_page.content.decode("ISO-8859-1")
    soup = BeautifulSoup(content)
    external_links = soup.find_all('p')
    for ext in external_links:
        if ext.text.strip()[:30] == 'Testo integrale della sentenza':
            return(ext.a.attrs['href'])
    return None

if __name__ == '__main__':
    session = HTMLSession()
    #initial URL
    starting_page = "https://www.giustiziatributaria.gov.it/gt/web/guest/archivio-sentenze-2020"
    # get the links to the sentences
    initial_page = requests.get(starting_page)
    content = initial_page.content.decode('utf-8')
    soup = BeautifulSoup(content)
    sentence_divs = soup.find_all('p',class_ = "par_sent")
    sentence_links = [sentence.a.attrs['href'] for sentence in sentence_divs]
    sentence_names = [sentence.a.text.strip().replace('/','') for sentence in sentence_divs]
    # get intermediate pages
    urls = [get_complete_sentence_link(url) for url in tqdm(sentence_links)]
    urls = [url for url in urls if url is not None]

    #for each document, extract and save text in .txt files
    for url,name in tqdm(zip(urls,sentence_names)):
        try:
            out = render_JS(url,session)
            soup = BeautifulSoup(out)
            div = soup.find_all('div', id = 'dettaglio-giurisprudenza
   ')

            with open(str(name)+'.txt','w',encoding="ISO-8859-1") as
   file:
                for d in div[0].find_all('p'):
                    out = " ".join([str(s).replace('<span>','').
   replace('</span>','') for s in d.contents if s is not None])
                    file.write(out+'\n')
        except Exception:
            print(url,name)
```

Code 3.1: Scraper implementation

After the documents are collected, all Hypertext References (HREFs) of laws need to be captured and marked in order to where the entity is located. HREF are always instantiated by the syntax *<a href=* and closed by <\/*a*>. An example of references is the following one:

<a href=d̈ecodeurn?urn=urn:doctrib::CPC:;_art132\Ξart.132c.p.c. < \/a >

A simple regular expression[9] locates all possible HREF (for each document) and two fictitious tokens, START_LEX and END_LEX, are placed respectively at the beginning and at the end of the laws references (these two tokens will not be saved inside the final annotated dataset). These two placeholders are used in the final part of the process. Each word is taken singularly and marked with its correspondent tag. In this case, we are only considering LEX (indicating laws references) and O (no entity) tags as possible choices. The algorithm takes word per word and labels them as O unless START_LEX is not encountered. At that point, next word is labeled as B-LEX and following ones with I-LEX until the END_LEX token.

```
    import re
from glob import glob

if __name__ == '__main__':

  for file_name in glob("C:/Users/rober/Desktop/2017/*"):
    file_name = file_name.replace("\\", "/")
```

[9]A regular expression is an encoded sequence of characters that specifies a search pattern inside texts

```
  print(file_name)
  with open(file_name, "r", errors = "ignore") as f:
    #documents location
    dataset_name = file_name.replace("C:/Users/rober/Desktop/2017/", "")
    #create spaces between punctuations to split correctly all the tokens
    with open(dataset_name, "w", errors = "ignore") as f1:
      line = f.read()
      line = line.replace(".", " . ")
      line = line.replace(":", " : ")
      line = line.replace(";", " ; ")
      line = line.replace(",", " , ")
      line = line.replace("-", " - ")
      line = line.replace("(", " ( ")
      line = line.replace(")", " ) ")
      print(line)

      entity_count = 0
      while(1):
        try:
          #regular expression for href detection
          refined_result = re.search(r"<a href.+?>", line)
          remove_part = line[refined_result.span()[0]:refined_result.span()[1]]
          #add fictitious tokens
          line = line.replace(remove_part, " START_LEX ", 1)
          line = line.replace("</a>", " END_LEX ", 1)
          entity_count += 1
        except:
          break

      text_tokens = line.split()

      i=0
      words_per_phrase = 35
      tag = "O"
      #cycle for automatic annotation of the dataset
      while(i < len(text_tokens)):
        if(text_tokens[i] == "START_LEX"):
          words_per_phrase = 35
          i-=15
          while(words_per_phrase >= 0):

              if(text_tokens[i] == "START_LEX"):
                tag = "B-LEX"


              elif(text_tokens[i] == "END_LEX"):
                tag = "O"


              elif(text_tokens[i] != "START_LEX" and text_tokens[i] != "END_LEX"):
                f1.write(str(text_tokens[i]) + " " + tag + "\n")
                if(tag == "B-LEX"):
                  tag = "I-LEX"


              i += 1
              words_per_phrase -= 1
              if(words_per_phrase < 0):
                if(tag == "B-LEX" or tag == "I-LEX"):
                  words_per_phrase += 7

              if(i==len(text_tokens)-1):
                words_per_phrase = 0

            f1.write("\n")

          i += 1
```

Code 3.2: Automatic algorithm for creating the annotated dataset

The complete procedure creates the final automatically annotated silver dataset. Because of the high numbers of available documents (see Table 3.9), they have been split in two sets: one for sentences containing LEX references and one containing only O tags. The two sets are finally mixed together, generating train and test set. Section This method clearly create some artifacts (e.g. some other HTML commands such as <p> to define paragraphs that should be manually checked and removed) in low percentages, but they should not visually compromise evaluations.

	Dataset LEX	Dataset O
# Documents	692	590
# Sentences	7,663	7184
# LEX	11,156	0

Table 3.9: LEX and O Dataset statistics

3.2.7 Mixed Dataset

Last part of the project, related to Transfer Learning, required multilingual dataset to verify the effectiveness of multilingual models on extracting context knowledge over a fixed set of entities in a source language different from the target language (for this work, target language is always Italian). Dataset presented in this chapter has been mixed to create multilingual modified corpus.

Legal-NER modified versions The final experiments of this project, related to public administration entities detection, wanted to ensure how the addition of Italian sentences inside the training set influences the overall performances (over predictions on the Italian artificial test set). Starting from the basic version of the Legal-NER dataset, a high number of different versions have been created. These modified versions are:

- **train_modified** - exact copy of the training set of Legal-NER, where the law entity GS is simply replaced by the entity LEX. This is done to maintain the entity tag coherence between all dataset involved in the experiments.
- **train_modified_only_LEX** - alternative version where all entities, except for LEX, are discarded and marked with the O label.
- **train_modified_50_ita** - train_modified version containing 50 Italian sentences taken from the artificial dataset. All the Italian sentences contains at least one reference to a law.
- **train_modified_100_ita** - train_modified version containing 100 Italian sentences taken from the artificial dataset. All the Italian sentences contains at least one reference to a law.
- **legal_train_with_ita** - Legal-NER training set containing about 5000 Italian sentences, 2500 with at least one law references and 2500 without LEX entity.
- **legal_valid_deu** - Legal-NER validation set with the already mentioned substitution of the tag GS with LEX
- **legal_valid_deu_only_LEX** - Legal-NER validation set containing only LEX and O tags

- **legal_test_deu** - Legal-NER test set again with the GS/LEX replacing
- **legal_test_deu_only_LEX** - Legal-NER test set containing only LEX and O tags
- **legal_test_ita** - final test set version containing only Italian sentences. It is composed by the REDIT data, plus other sentences (with and without law references) taken from the artificial dataset

4 Experiments and Analysis

4.1 Introduction

This section completely covers all the strategies and experiments done with dataset and models quoted in previous chapters. First part will be mainly focused on analyzing and understanding which model is the most suitable for the characteristic of this project. Second part will explore the Transfer Learning techniques and it will go into detail especially on the modification of some parameters of the model architecture. Experiments have been led both in Google Colab[1] or in AWS[2] (Amazon Web Services) to exploit their possibility to use GPU in runtime and higher amounts of RAM and disk space.

4.1.1 Evaluations metric: F1-score

Evaluations on the predictions made from the models can be done in many ways. One of the most representative (based on the collection of papers related to the NER task) is to compute the F1-score. F1-score is a metric for test accuracy based on two different values:

- *Precision* - It is the number of true positive instances divided by the number of all positive results.
- *Recall* - It is the number of true positive results divided by all that instances that should have been identified as positive

[1]https://colab.research.google.com/
[2]AWS is a platform that offers over 200 services for cloud computing

It is computed as the harmonic mean of precision and recall. The final evaluation is per entity and not per token.

$$F1 - score = \frac{2 * precision * recall}{precision + recall} \tag{4.1}$$

F1-score ranges from 0 (bad performances) to 1 (good performances). In case of multi-label case the final score is computed as the average, that can be differentiated in three types:

- *micro* - F1-score is computed globally, counting total true positives, false negatives and false positive instances
- *macro* - F1-score is calculated as the average of the F1-score of each single label. It does not take into account the label imbalance
- *weighted* - F1-score is computed as the average of the F1-score of each single label, weighted by their relative support (number of true positive instances)

There is not a fixed rule to decide which average type is the best one, but it depends explicitly on the context of each situation.

4.1.2 Error Analysis

F1-score is a really effective metric to represent the accuracy of a model, but it has some weaknesses. The main reason is correlated to the fact that F1-score does not weight all the possible errors that a model can commit. They can be divided in four classes:

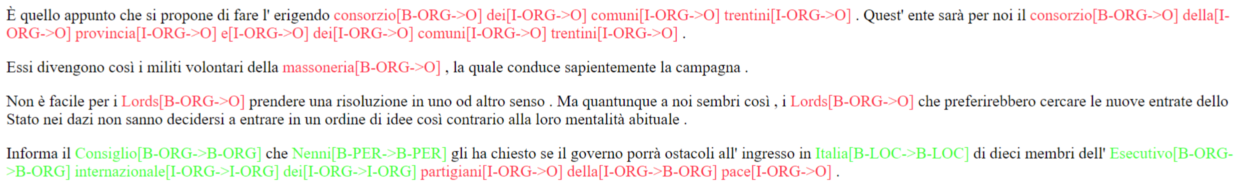
- **missed entity** - the model does not recognize the entity and each token is tagged as O (outside).
- **wrong tag** - the entity is captured correctly, but with the wrong tag. This error is usually detected inside that type of entities that are not uniquely defined inside the dataset. For instance, inside the KIND dataset, we can find references to states such as Italy or Austria identified sometimes with ORG and sometimes with LOC tag, based on the context of the sentences.
- **O tag identified as a label** - in this case a sequence of O tags is identified as an existing entity
- **span errors** - An entity is found, but not with the correct span (in addition, the tag error could be correct or wrong)

The definition of these classes is anyway borderline, since from the context of each sentence can be derived a different views of each possible type of error (i.e. errors have different seriousness with respect to the context). Because of this reason, it is necessary to analyze manually the dataset and decide a level of tolerance for each possible class. A piece of code has been developed to detect all the pair of gold/predicted tag (for each token). All the sentences containing one or more possible errors are saved inside an HTML files (one file per entity), where correct pairs are highlighted in green, while wrong pairs are marked in red.

```python
if __name__ == "__main__":
  with open("prediction_lex.txt", "r", encoding = "utf-8") as f:
    lines = f.readlines()

    #labels contained in the dataset
    #labels = ["O", "B-Organization", "I-Organization", "B-Location",
     "I-Location", "B-Profession", "I-Profession", "B-Name", "I-Name",
     "B-Contact", "I-Contact"]
    #labels = ["O", "B-ORG", "I-ORG", "B-LOC", "I-LOC", "B-PER", "I-
   PER"]
    labels = ["O", "B-LEX", "I-LEX"]
    for label in labels:
      total = 0
      correct = 0
      wrong = 0
      #dictionary counting all the errors for each entity
      #dict_count = {"O": 0, "B-Organization": 0, "I-Organization":
   0, "B-Location": 0, "I-Location": 0, "B-Profession": 0, "B-Name":
   0, "I-Name": 0, "I-Profession": 0, "B-Contact": 0, "I-Contact": 0}
      #dict_count = {"O": 0, "B-ORG": 0, "I-ORG": 0, "B-LOC": 0, "I-
   LOC": 0, "B-PER": 0, "I-PER": 0}
      dict_count = {"O": 0, "B-LEX": 0, "I-LEX": 0}
      wrong_words = []
      complete_lines = []
      for line in lines:
        tokens = line.split()
        if(line != "\n"):
          if(tokens[2] == label):
            if(tokens[2] != tokens[1]):
              total += 1
              wrong += 1
              dict_count[tokens[1]] += 1
              wrong_words.append(tokens[0])
              complete_lines.append(line)

        else:
          total += 1
          correct += 1


  #order the dictionary
  ordered_dict_count = {k: v for k, v in sorted(dict_count.items
(), key=lambda item: item[1], reverse = True)}

  #print statistics
  print("LABEL: {}\n".format(label))
  print("Total occurrences: {}".format(total))
  print("Correct occurrences: {}".format(correct))
  print("Wrong occurrences: {}".format(wrong))
  print("Errors made:\n{}\n".format(ordered_dict_count))
  print("Word errors:\n{}\n".format(wrong_words))
  print("\n\n")


  #save sentences inside html file
  set_lines = set(complete_lines)
  file_name = "predicted_" + str(label) + ".html"
  with open(file_name, "w", encoding="utf-8") as f:
    f.write("<p>\n")
    f.write("GOLD LABEL: " + str(label) +"\n\n")
    f.write("<br> <br>")
    f.write("WRONG WORDS:\n" + str(set_lines) + "\n\n")
    for complete_line in set_lines:
      i = 0
      f.write("<p>\n")
      while(i < len(lines)):

        if(lines[i] == complete_line):
          while(lines[i-1] != "\n"):
            i -= 1

          while(lines[i] != "\n"):
            tokens = lines[i].split()

            if(tokens[2] != tokens[1]):
              #f.write(str(tokens[0]) + "[" + str(tokens[1]) +
"->" + str(tokens[2]) + "] ")
              f.write("<span style=\"color: #FF334C\">" + str(
tokens[0]) + "[" + str(tokens[1]) + "->" + str(tokens[2]) + "] " +
 "</span>")

                elif(tokens[1] != "O"):
                  f.write("<span style=\"color: #39FF33\">" + str(
tokens[0]) + "[" + str(tokens[1]) + "->" + str(tokens[2]) + "] " +
 "</span>")

                else:
                  f.write(str(tokens[0]) + " ")

                i += 1

              f.write("\n\n")
            i += 1
          f.write("</p>")
```

Code 4.1: Error analysis code

Some statistics are printed as output in the terminal, containing useful information such as total, correct and wrong occurrences and an ordered dictionary (descending order) indicating the number of times each gold label has been wrongly predicted (Fig 4.1).
This complete procedure asks anyway a manual revision over the entire set of HTML files, that results too expensive in term of times. For this reason, only few section of tests will cover this part.


È quello appunto che si propone di fare l' erigendo consorzio[B-ORG->O] dei[I-ORG->O] comuni[I-ORG->O] trentini[I-ORG->O] . Quest' ente sarà per noi il consorzio[B-ORG->O] della[I-ORG->O] provincia[I-ORG->O] e[I-ORG->O] dei[I-ORG->O] comuni[I-ORG->O] trentini[I-ORG->O] .

Essi divengono così i militi volontari della massoneria[B-ORG->O] , la quale conduce sapientemente la campagna .

Non è facile per i Lords[B-ORG->O] prendere una risoluzione in uno od altro senso . Ma quantunque a noi sembri così , i Lords[B-ORG->O] che preferirebbero cercare le nuove entrate dello Stato nei dazi non sanno decidersi a entrare in un ordine di idee così contrario alla loro mentalità abituale .

Informa il Consiglio[B-ORG->B-ORG] che Nenni[B-PER->B-PER] gli ha chiesto se il governo porrà ostacoli all' ingresso in Italia[B-LOC->B-LOC] di dieci membri dell' Esecutivo[B-ORG->B-ORG] internazionale[I-ORG->I-ORG] dei[I-ORG->I-ORG] partigiani[I-ORG->O] della[I-ORG->B-ORG] pace[I-ORG->O] .

Figure 4.1: Example of error analysis HTML format

4.2 Experiments - First part

First experiments are much more focused on obtaining a strong theoretical background of models, extracting every possible information useful to understand the better choice of the model for the next steps of this project.

4.2.1 CRF

CRF model has been proven to be really effective when is placed as final layer over a stronger model. But it can also work alone and extract features context over the data and, with quite good accuracy, return predictions over the test set. The idea is to verify and compare its performances with respect to the other models considered. Similar implemented code[3] have been exploited and modified to support the right reading of the CONLL2003 dataset. The training is made using the Limited-memory BFGS (L-BFGS) (that is set as default), a popular algorithm for parameter estimation. The F1-score is used as evaluation method.

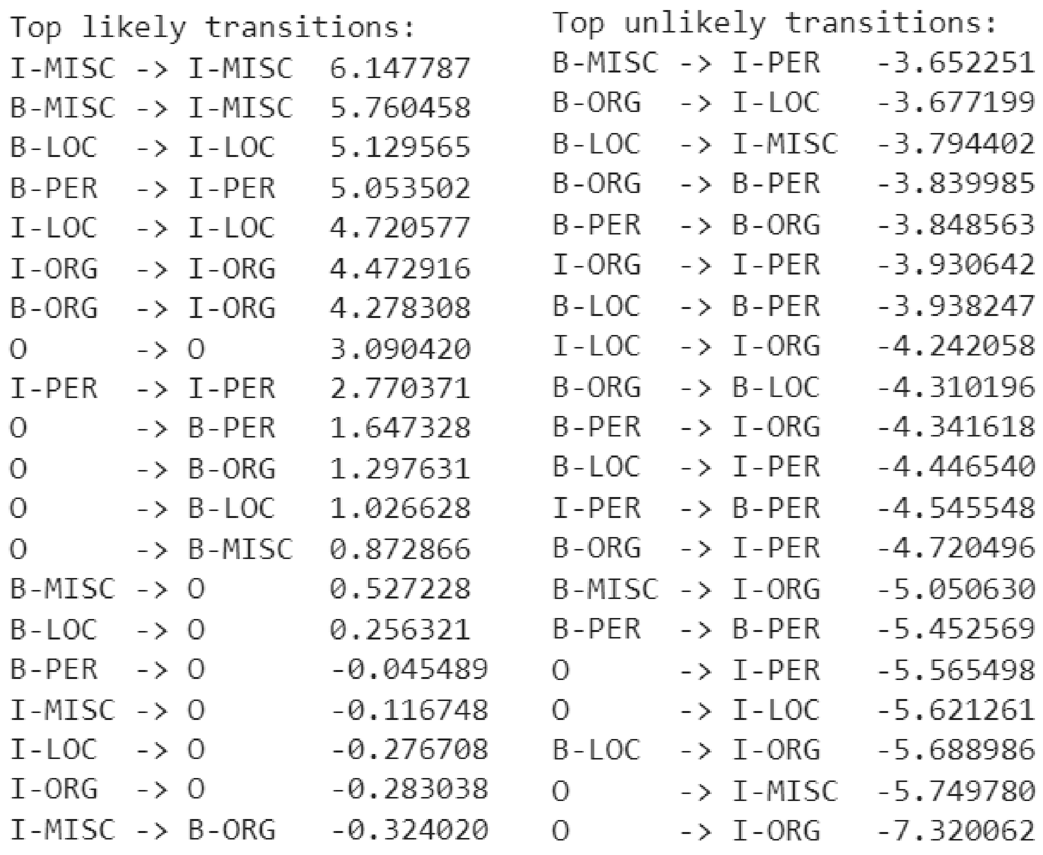
	precision	recall	f1-score	support
B-LOC	0.8531	0.8076	0.8297	1668
I-LOC	0.7536	0.6187	0.6795	257
B-MISC	0.8110	0.7578	0.7835	702
I-MISC	0.6729	0.6667	0.6698	216
B-ORG	0.7622	0.7255	0.7434	1661
I-ORG	0.6696	0.7353	0.7009	835
B-PER	0.8250	0.8541	0.8393	1617
I-PER	0.8645	0.9490	0.9047	1156
micro avg	0.7998	0.7987	0.7992	8112
macro avg	0.7765	0.7643	0.7688	8112
weighted avg	0.8000	0.7987	0.7983	8112

Table 4.1: CRF - F1-score

Table 4.1 shows quite good performances, considering that CRF is a very light-weight model (1.5 MB) and requires a low amount of resources for training (less than a minute) and evaluation (just few seconds). Its strength is linked to the ability on accurately detecting all the possible transitions between entities, learning the most frequent, but also transitions that will never appear (e.g. it is not possible to have and I tag after a O for BIO tagging). CRF gives high

[3]`https://sklearn-crfsuite.readthedocs.io/en/latest/` - this tutorial provide an implementation of CRF based on CONLL2002

positive scores for most frequent transition and negative scores for the ones that are usually discarded or directly cannot be used considering the BIO tagging grammar (e.g. it is not possible to see a transition from O-tag to I-tag, or from a B-tag to I-tag of two different entities).



```
Top likely transitions:          Top unlikely transitions:
I-MISC -> I-MISC  6.147787       B-MISC -> I-PER   -3.652251
B-MISC -> I-MISC  5.760458       B-ORG  -> I-LOC   -3.677199
B-LOC  -> I-LOC   5.129565       B-LOC  -> I-MISC  -3.794402
B-PER  -> I-PER   5.053502       B-ORG  -> B-PER   -3.839985
I-LOC  -> I-LOC   4.720577       B-PER  -> B-ORG   -3.848563
I-ORG  -> I-ORG   4.472916       I-ORG  -> I-PER   -3.930642
B-ORG  -> I-ORG   4.278308       B-LOC  -> B-PER   -3.938247
O      -> O       3.090420       I-LOC  -> I-ORG   -4.242058
I-PER  -> I-PER   2.770371       B-ORG  -> B-LOC   -4.310196
O      -> B-PER   1.647328       B-PER  -> I-ORG   -4.341618
O      -> B-ORG   1.297631       B-LOC  -> I-PER   -4.446540
O      -> B-LOC   1.026628       I-PER  -> B-PER   -4.545548
O      -> B-MISC  0.872866       B-ORG  -> I-PER   -4.720496
B-MISC -> O       0.527228       B-MISC -> I-ORG   -5.050630
B-LOC  -> O       0.256321       B-PER  -> B-PER   -5.452569
B-PER  -> O       -0.045489      O      -> I-PER   -5.565498
I-MISC -> O       -0.116748      O      -> I-LOC   -5.621261
I-LOC  -> O       -0.276708      B-LOC  -> I-ORG   -5.688986
I-ORG  -> O       -0.283038      O      -> I-MISC  -5.749780
I-MISC -> B-ORG   -0.324020      O      -> I-ORG   -7.320062
```

Figure 4.2: Top likely/unlikely transition for CRF on CONLL2003

These results gives an idea on why models in general benefits by adding a CRF layer, since it is capable to further increase performances with negligible need of resources.

4.2.2 BI-LSTM-CRF

After the examination of CRF, the next step lead to the analysis of the Bi-LSTM-CRF. The model analysis is quite more complex in this case, since the final evaluation is affected by much more parameters rather than in the case of CRF. Parameters that has been considered are:

- character-level embedding dimension: set as 25 as default (total dimension is then 400d, since the LSTM layers are bi-directional)
- word embeddings: Glove for English dataset (CONLL2003 and Jobstack) and Italian FastText word embeddings for KIND. For words in text documents that are not contained inside Glove or FastText, a randomized word embedding vector (of same dimension) is assigned.
- word embedding dimension: Glove embeddings can be in 100d, 200d and 300d format, while FastText are available only in 300d
- layers LSTM dimension: the default value is 200d (total dimension is then 400d, since the LSTM layers are bi-directional)
- number of epochs for training phase

Next tables highlight some of the best results with a final summary of all the evaluations. To distinguish between them, a specific name format is assigned, that is *dataset_character-level-dim_embeddings-dim_layers-dim* (e.g. jobstack_25d_100d_200d). Table 4.5, to give a general view, considers only weighted average, due to the high unbalance of support values between all the entities (this makes macro average quite unreliable).

	precision	recall	f1-score	support
B-ORG	0.8333	0.6286	0.7166	350
B-LOC	0.7854	0.7478	0.7661	230
I-ORG	0.9298	0.5792	0.7138	183
B-PER	0.8235	0.7962	0.8096	211
I-PER	0.9189	0.8095	0.8608	42
I-LOC	0.8095	0.3864	0.5231	44
macro avg	0.8500	0.6580	0.7540	1060
weighted avg	0,8400	0,6764	0,7582	1060

Table 4.2: KIND_25d_300d_350d (10 epochs)

	precision	recall	f1-score	support
B-LOC	0.9043	0.8951	0.8997	1668
B-PER	0.8972	0.9066	0.9019	1617
B-ORG	0.8597	0.8266	0.8428	1661
I-PER	0.9412	0.9697	0.9553	1156
I-ORG	0.8228	0.8395	0.8311	835
B-MISC	0.7742	0.7863	0.7802	702
I-LOC	0.8127	0.7938	0.8031	257
I-MISC	0.6498	0.7130	0.6799	216
macro avg	0.8327	0.8412	0.8369	46435
weighted avg	0.8696	0.8708	0.8702	46435

Table 4.3: CONLL2003_25d_100d_100d (10 epochs

	precision	recall	f1-score	support
B-Organization	0.9371	0.7163	0.8120	208
B-Location	0.8897	0.9085	0.8990	142
I-Organization	0.8318	0.7876	0.8091	113
I-Location	0.9115	0.8512	0.8803	121
I-Profession	0.9100	0.8273	0.8667	110
B-Profession	0.8448	0.7656	0.8033	64
B-Contact	0.8571	0.8571	0.8571	7
B-Name	0.5000	0.6000	0.5455	5
I-Name	0.6000	0.6000	0.6000	5
I-Contact	0.0000	0.0000	0.0000	0
macro avg	0.7282	0.6914	0.7073	775
weighted avg	0.8919	0.8026	0.8425	775

Table 4.4: Jobstack_25d_300d_100d (10 epochs)

Name	embeddings	epochs	precision	recall	f1-score
jobstack_25d_300d_100d	GloVe	25	0.8549	0.8309	0.8397
jobstack_25d_200d_100d	GloVe	10	0.8653	0.7627	0.8042
jobstack_25d_200d_100d	GloVe	25	0.8674	0.7987	0.8294
CONLL2003_25d_300d_100d	**GloVe**	**15**	**0.8698**	**0.8706**	**0.8700**
CONLL2003_25d_200d_100d	GloVe	15	0.8623	0.8583	0.8596
CONLL2003_25d_300d_100d	GloVe	20	0.8606	0.8666	0.8632
KIND_25d_300d_100d	FastText	10	0.8142	0.7104	0.7566
KIND_25d_300d_100d	FastText	20	0.7811	0.6952	0.7319
jobstack_25d_300d_200d	**GloVe**	**15**	**0.8919**	**0.8026**	**0.8425**
jobstack_25d_300d_350d	GloVe	10	0.8479	0.7819	0.8094
KIND_25d_300d_350d	**FastText**	**10**	**0,8400**	**0,6764**	**0,7582**

Table 4.5: Summary of tests over Bi-LSTM-CRF model (weighted average)

Table 4.5 depicts the good average behaviour of the Bi-LSTM in different circumstances. For English datasets, all the various configurations confirm that the model does not require a high number of epochs (10/15 are enough) and size of the LSTM layers, since overextending these parameters lead to worst performances, due to over-fitting (the model is not able to generalize its knowledge, being too constrained to the train set). For Italian dataset, the model follows the same logic for the number of epochs, while increasing the size of LSTM layers seems to not improve the situation, since higher value lead to a better precision score (see Table 4.5), but a loss in terms of recall. In general, Bi-LSTM-CRF can be considered as a complex model with several advantages related to the amount of resources required for training and evaluation phase. It is pretty light-weight (none of the model goes over 150 MB) and does not strictly require GPU for processing.

Error Analysis The developed error analysis script is used here to understand how harsh is the computation of F1-score. Indeed, there may be a high number of cases where the prediction made by the model differs to the gold prediction. But as already mentioned, the division between different domains is sometimes borderline due to the context of sentences. Table 4.6 show a generic view of the errors, reporting the main statistics over prediction of Jobstack test set.
The manual observation of each HTML files indicates that many errors are often

Label	# occur.	Correct occur.	Wrong occur.
O	24478	24371	107
B-Organization	189	153	36
I-Organization	137	89	48
B-Location	158	132	26
I-Location	114	95	19
B-Profession	63	54	9
I-Profession	119	99	20
B-Name	6	3	3
I-Name	6	3	3
B-Contact	10	6	4
I-Contact	0	0	0

Table 4.6: Bi-LSTM error analysis

related to the a word that is repeated many times over the dataset. This is not surprising, since static embeddings offer the same vector representation of each word occurrence, leading the model to predict each word always with the same tag (independently from the context). Luckily, most of these errors are part of the borderline discussion or are related to the span, where the entity is correctly detected but with minimal mistakes on the tags interval. This means that the model performs better than what the F1-score reports.

4.2.3 BERT-CRF

First phase of the projects ends with computations on BERT-CRF. Considering state of the art performances, it surely outperforms Bi-LSTM in each general tasks, thanks to dynamic embeddings and a much more complex structure (with a higher number of trainable parameters). But, as we stated before, the objective is to understand if this power worth the total costs. Indeed, evaluations will be more accurate, but they have to be compared also with difference of the time and physical resources needed. For instance, this model always require GPU for training and prediction phase, otherwise the complexity of the architecture is not sustainable. Next tables shows all the generic results obtained for English and Italian tests.

	precision	recall	f1-score	support
B-LOC	0.92	0.94	0.93	1668
B-PER	0.98	0.97	0.97	1617
B-ORG	0.91	0.89	0.90	1661
I-PER	0.99	0.98	0.98	1156
I-ORG	0.87	0.89	0.88	835
B-MISC	0.81	0.84	0.82	702
I-LOC	0.82	0.90	0.86	257
I-MISC	0.64	0.77	0.70	216
macro avg	0.87	0.90	0.88	46435
weighted avg	0.91	0.92	0.92	46435

Table 4.7: BERT_CONLL_1

	precision	recall	f1-score	support
B-LOC	0.93	0.93	0.93	1668
B-PER	0.96	0.96	0.96	1617
B-ORG	0.90	0.92	0.91	1661
I-PER	0.98	0.99	0.99	1156
I-ORG	0.89	0.93	0.91	835
B-MISC	0.83	0.86	0.84	702
I-LOC	0.88	0.89	0.89	257
I-MISC	0.64	0.79	0.71	216
macro avg	0.88	0.91	0.89	46435
weighted avg	0.91	0.93	0.92	46435

Table 4.8: BERT_CONLL_2

	precision	recall	f1-score	support
B-Organization	0.82	0.79	0.81	208
B-Location	0.88	0.96	0.92	142
I-Organization	0.78	0.81	0.80	113
I-Location	0.87	0.87	0.87	121
I-Profession	0.84	0.91	0.87	110
B-Profession	0.80	0.88	0.84	64
B-Contact	0.86	0.86	0.86	7
B-Name	0.83	1.00	0.91	5
I-Name	0.83	1.00	0.91	5
I-Contact	0.00	0.00	0.00	0
macro avg	0.75	0.80	0.78	775
weighted avg	0.83	0.86	0.85	775

Table 4.9: BERT_jobstack_1

	precision	recall	f1-score	support
B-Organization	0.82	0.86	0.84	208
B-Location	0.88	0.93	0.90	142
I-Organization	0.79	0.82	0.81	113
I-Location	0.83	0.88	0.86	121
I-Profession	0.91	0.88	0.89	110
B-Profession	0.82	0.86	0.84	64
B-Contact	0.86	0.86	0.86	7
B-Name	0.83	1.00	0.91	5
I-Name	0.83	1.00	0.91	5
I-Contact	0.00	0.00	0.00	0
macro avg	0.75	0.80	0.78	775
weighted avg	0.84	0.87	0.86	775

Table 4.10: BERT_jobstack_2

	precision	recall	f1-score	support
B-ORG	0.77	0.73	0.75	350
B-LOC	0.88	0.85	0.87	230
I-ORG	0.76	0.80	0.78	183
B-PER	0.93	0.94	0.94	211
I-PER	0.97	0.90	0.94	42
I-LOC	0.85	0.75	0.80	44
macro avg	0.86	0.82	0.85	1060
weighted avg	0,84	0,81	0,83	1060

Table 4.11: BERT_KIND

As expected, BERT-CRF has proven to be really effective, both in English and in Italian. Considering all the three datasets, it shows an average improvement (weighted F1-score is considered) of 5%, with respect to Bi-LSTM (from 82& to 87%). This is surely not negligible, but computation times of training and evaluations are doubled. This is a critical point, where users has to define, based on necessity of the project, which of the two models satisfies better all its conditions.

4.3 Experiments - Second part

First part of the experiments lead to the definition of optimal models for prediction of privacy-related entities. They can be defined as canonical, since we consider classical entities related to NER. The advantage is that it is possible to find an infinite number of possible datasets, freely available in the internet. However, approaching de-identification means also encountering other non-canonical entities, where the amount of related data is no more ensured. The first class to be tested is the quotes of laws inside documents, identified with LEX tag. The only Italian dataset obtained is REDIT (Paccosi and Palmero, 2021 [11]), but unfortunately not the complete version (copyright issues), not sufficient to produce a significant training set. Subsequent researches discovered Legal-NER dataset and the archive of "Ministero dellEconomia e delle Finanze", already presented in chapter 3. Based on these resources, the idea to deal with all the problems is to exploit the high flexibility of BERT, that can be trained and produce multilingual model, able to make prediction over different languages. This is linked to Transfer Learning, a research problem in Machine Learning, where knowledge stored by a model for facing up to a specific problem is applied to a second different (but related) task. The expectation is that, training multilingual-BERT to identify laws in a source language, it will make it able to provide good prediction also in the target language. Transfer Learning is still a developing branch of Artificial Intelligence. That's why foreplay monolingual experiments using multilingual models has been processed, to ensure the general efficiency of them. In addition, to improve general performances, some layers have been frozen. The idea is that fine-tuning the complete architecture will lead to an overfitting related to the source language, causing the model to forget its previous knowledge over the other set of languages. Freezing some layers, instead, makes this process less aggressive, exploiting fine-tuning just to specialize BERT over NER task, while maintaining its general multilingual knowledge. Finally, number of epochs is kept fixed to 3. Higher numbers, in general, cause again overfitting, leading always to lower evaluation scores.

4.3.1 Multilingual-BERT: Monolingual Experiments

As anticipated, these tests aim to guarantee a basic functionality of these type of models, over monolingual train and test set. The most important thing over these experiments is also to understand the efficiency of CRF: Indeed, this layer is fundamentally used to understand the most probable sequence of tags inside entities. This means that, unconsciously, it learns also the grammar of the source language used in the training set. Dealing with different languages can cause a variation over their grammar rules, where the sequence of verbs, nouns and adjectives can be different. This clearly cannot produce a good general context, leading to a split of test for multilingual BERT, both with and without CRF.

English-English Related to laws identification, English language is not that useful for this project, since no English datasets have been used at this purpose. Its main utility is to get a comparison with scores obtained over CONLL2003 in the first part of the experiments, understanding how close are the evaluations for the multilingual model. Next tables list the best results obtained. Every test name will have the following format: *m-BERT_flag-CRF_num-layers-frozen*

	precision	recall	f1-score	support
B-LOC	0.93	0.93	0.93	1668
B-PER	0.96	0.96	0.96	1617
B-ORG	0.87	0.92	0.90	1661
I-PER	0.99	0.98	0.98	1156
I-ORG	0.87	0.92	0.90	835
B-MISC	0.84	0.84	0.84	702
I-LOC	0.85	0.90	0.87	257
I-MISC	0.64	0.76	0.70	216
macro avg	0.87	0.90	0.88	46435
weighted avg	0.91	0.92	0.91	46435

Table 4.12: m-BERT_No-CRF_0 (English-English)

	precision	recall	f1-score	support
B-LOC	0.93	0.93	0.93	1668
B-PER	0.96	0.96	0.96	1617
B-ORG	0.89	0.91	0.90	1661
I-PER	0.98	0.99	0.98	1156
I-ORG	0.88	0.92	0.90	835
B-MISC	0.82	0.85	0.84	702
I-LOC	0.84	0.91	0.87	257
I-MISC	0.64	0.77	0.70	216
macro avg	0.87	0.90	0.88	46435
weighted avg	0.91	0.92	0.91	46435

Table 4.13: m-BERT_No-CRF_4 (English-English)

	precision	recall	f1-score	support
B-LOC	0.93	0.93	0.93	1668
B-PER	0.97	0.95	0.96	1617
B-ORG	0.89	0.92	0.91	1661
I-PER	0.98	0.99	0.99	1156
I-ORG	0.88	0.93	0.90	835
B-MISC	0.82	0.85	0.83	702
I-LOC	0.86	0.93	0.89	257
I-MISC	0.70	0.76	0.72	216
macro avg	0.88	0.91	0.89	46435
weighted avg	0.91	0.93	0.92	46435

Table 4.14: m-BERT_CRF_0 (English-English)

	precision	recall	f1-score	support
B-LOC	0.94	0.93	0.93	1668
B-PER	0.97	0.96	0.96	1617
B-ORG	0.89	0.92	0.90	1661
I-PER	0.99	0.99	0.99	1156
I-ORG	0.88	0.92	0.90	835
B-MISC	0.82	0.87	0.84	702
I-LOC	0.85	0.92	0.88	257
I-MISC	0.70	0.77	0.73	216
macro avg	0.88	0.91	0.89	46435
weighted avg	0.92	0.93	0.92	46435

Table 4.15: m-BERT_CRF_4 (English-English)

As expected, CRF gives a small (about 1%) improvement. Considering the already high evaluation scores, also minimal upgrade must be considered. Anyway, the most important thing is that multilingual-BERT, operating in monolingual world, behaves exactly as the BERT-CRF model presented in the previous section. This is a critical point, since we are sure that, considering multilingual test, there will be no loss on efficiency due to the model itself.

German-German Similarly to the previous paragraph, these tests aim to verify the right processing of the model over German language. This is the first important point that will lead to the de-identification of laws. As before, next tables list the general results with the same name format.

	precision	recall	f1-score	support
B-LOC	0.78	0.78	0.78	1024
B-PER	0.93	0.91	0.92	1193
B-ORG	0.77	0.60	0.68	773
I-PER	0.97	0.97	0.97	631
I-ORG	0.80	0.80	0.80	489
I-LOC	0.81	0.53	0.64	261
macro avg	0.84	0.76	0.80	46435
weighted avg	0.85	0.80	0.82	46435

Table 4.16: m-BERT_No-CRF_0 (German-German)

	precision	recall	f1-score	support
B-LOC	0.80	0.79	0.79	1024
B-PER	0.91	0.92	0.91	1193
B-ORG	0.82	0.57	0.68	773
I-PER	0.96	0.98	0.97	631
I-ORG	0.86	0.69	0.77	489
I-LOC	0.80	0.63	0.71	261
macro avg	0.84	0.76	0.80	46435
weighted avg	0.85	0.80	0.82	46435

Table 4.17: m-BERT_No-CRF_4 (German-German)

	precision	recall	f1-score	support
B-LOC	0.82	0.80	0.81	1024
B-PER	0.93	0.89	0.91	1193
B-ORG	0.82	0.59	0.69	773
I-PER	0.97	0.97	0.97	631
I-ORG	0.83	0.73	0.78	489
I-LOC	0.81	0.62	0.70	261
macro avg	0.86	0.77	0.81	46435
weighted avg	0.87	0.79	0.83	46435

Table 4.18: m-BERT_CRF_0 (German-German)

	precision	recall	f1-score	support
B-LOC	0.81	0.79	0.80	1024
B-PER	0.92	0.90	0.91	1193
B-ORG	0.81	0.59	0.69	773
I-PER	0.98	0.98	0.98	631
I-ORG	0.84	0.77	0.80	489
I-LOC	0.76	0.64	0.70	261
macro avg	0.85	0.78	0.81	46435
weighted avg	0.86	0.80	0.83	46435

Table 4.19: m-BERT_CRF_4 (German-German)

These scores confirms again all the expectation: CRF, even in low percentage, helps the model to better predict the correct sequence of tags inside sentences (1% as for English-English).

4.3.2 Multilingual-BERT: Multilingual Experiments

Monolingual experiments confirms the efficiency of multilingual-BERT architecture. The next phase is to apply Transfer Learning and understand if the loss in term of evaluation is tolerable. First hypothesis suppose that, with respect to evaluation on monolingual tests, the percentage loss should be around 10%. Again, both versions of CONLL2003 are used as training set, while for prediction KIND test set has been chosen.

English-Italian This comparison is again not linked directly to the de-identification of laws, but to ensure the power of the model over Transfer Learning with different languages. Next tables show the best results over this experiment, using the usual name format.

	precision	recall	f1-score	support
B-ORG	0.78	0.69	0.73	350
B-LOC	0.72	0.89	0.80	230
I-ORG	0.72	0.70	0.71	183
B-PER	0.83	0.81	0.82	211
I-PER	0.61	0.64	0.62	42
I-LOC	0.52	0.40	0.46	44
macro avg	0.70	0.68	0.69	1060
weighted avg	0,75	0,74	0,74	1060

Table 4.20: m-BERT_No-CRF_0 (English-Italian)

	precision	recall	f1-score	support
B-ORG	0.78	0.69	0.73	350
B-LOC	0.74	0.88	0.80	230
I-ORG	0.73	0.68	0.71	183
B-PER	0.87	0.87	0.87	211
I-PER	0.77	0.83	0.80	42
I-LOC	0.62	0.40	0.50	44
macro avg	0.75	0.73	0.74	1060
weighted avg	0,77	0,76	0,76	1060

Table 4.21: m-BERT_No-CRF_4 (English-Italian)

	precision	recall	f1-score	support
B-ORG	0.77	0.64	0.70	350
B-LOC	0.73	0.82	0.77	230
I-ORG	0.66	0.64	0.65	183
B-PER	0.81	0.82	0.81	211
I-PER	0.75	0.78	0.76	42
I-LOC	0.33	0.40	0.36	44
macro avg	0.67	0.68	0.67	1060
weighted avg	0,73	0,71	0,72	1060

Table 4.22: m-BERT_CRF_0 (English-Italian)

	precision	recall	f1-score	support
B-ORG	0.79	0.66	0.72	350
B-LOC	0.72	0.85	0.78	230
I-ORG	0.67	0.68	0.67	183
B-PER	0.85	0.85	0.85	211
I-PER	0.91	0.78	0.84	42
I-LOC	0.42	0.44	0.43	44
macro avg	0.72	0.71	0.71	1060
weighted avg	0,75	0,74	0,74	1060

Table 4.23: m-BERT_CRF_4 (English-Italian)

The outcome confirms again the general initial considerations: the model does not perform as well as a monolingual one, but the general loss is below the 10%. This highlights the high capability of understanding the general context even between different languages, and apply it over NER task. Moreover, removing CRF layer and freezing some layers seems beneficial over multilingual condition (with about a 4% improvement), lightening the overfitting problem.

German-Italian These are approximately directly linked experiments to the real objective of the project. It is important to try to maximize general score over this test, since it would be a critical point to define a good general model that can work using Transfer Learning over every possible condition and entity encountered. As usual, next tables list the most interesting results.

	precision	recall	f1-score	support
B-ORG	0.79	0.26	0.39	350
B-LOC	0.70	0.87	0.77	230
I-ORG	0.76	0.52	0.63	183
B-PER	0.92	0.79	0.85	211
I-PER	0.93	0.60	0.73	42
I-LOC	0.60	0.34	0.43	44
macro avg	0.78	0.56	0.63	1060
weighted avg	0,79	0,56	0,62	1060

Table 4.24: m-BERT_No-CRF_0 (German-Italian)

	precision	recall	f1-score	support
B-ORG	0.80	0.45	0.58	350
B-LOC	0.75	0.86	0.80	230
I-ORG	0.84	0.53	0.65	183
B-PER	0.86	0.86	0.86	211
I-PER	0.81	0.75	0.78	42
I-LOC	0.63	0.38	0.48	44
macro avg	0.78	0.64	0.69	1060
weighted avg	0,80	0,64	0,70	1060

Table 4.25: m-BERT_No-CRF_4 (German-Italian)

	precision	recall	f1-score	support
B-ORG	0.70	0.16	0.26	350
B-LOC	0.72	0.84	0.78	230
I-ORG	0.81	0.36	0.50	183
B-PER	0.85	0.79	0.82	211
I-PER	0.84	0.58	0.68	42
I-LOC	0.46	0.38	0.42	44
macro avg	0.73	0.52	0.58	1060
weighted avg	0,75	0,49	0,55	1060

Table 4.26: m-BERT_CRF_0 (German-Italian)

	precision	recall	f1-score	support
B-ORG	0.77	0.25	0.37	350
B-LOC	0.73	0.88	0.79	230
I-ORG	0.85	0.36	0.50	183
B-PER	0.92	0.87	0.89	211
I-PER	1.00	0.75	0.86	42
I-LOC	0.50	0.43	0.46	44
macro avg	0.79	0.59	0.64	1060
weighted avg	0,80	0,55	0,61	1060

Table 4.27: m-BERT_CRF_4 (German-Italian)

In this case, the model with CRF layer fails completely the task, with a general loss that exceeds too much the hypothetical 10% tolerance. Comparing to the English-Italian tests, the scores are evidently worse. A possible consideration is the following one: English and Italian are generically much more similar languages, both morphologically and grammatically, making much easier the job for the model on understanding the context. On the other side, these similitudes are not so detectable between German and Italian, leading to a not useful application of CRF. Anyway, the freezing trick (in addition to the removal of CRF) seems to be very effective, mitigating that general overfitting problem.

Error Analysis At this point, a brief error analysis is needed to understand if there are some critical points related to the architecture of the model or if it is possible again to find many situations where predicted error tags are not so relevant. Tables 4.28 and 4.29 show generic statistics on English and German tests over KIND dataset.

Label	# occur.	Correct occur.	Wrong occur.
O	25923	25810	113
B-ORG	186	145	41
I-ORG	111	96	15
B-LOC	285	201	84
I-LOC	32	16	16
B-PER	178	166	12
I-PER	34	28	6

Table 4.28: m-BERT error analysis (English)

Label	# occur.	Correct occur.	Wrong occur.
O	25923	25986	274
B-ORG	186	145	36
I-ORG	137	89	48
B-LOC	158	132	26
I-LOC	114	95	19
B-PER	6	3	3
I-PER	6	3	3

Table 4.29: m-BERT error analysis (German)

In general, the major part of the mistakes are related to both the models. Furthermore, a high percentage of them can be easily discarded, since they do not result so relevant. For instance, a large set of errors are related to sentences containing quotes of States (Italy, Spain, France, etc..), where sometimes they are classified as ORG (indicating more precisely the government) and sometimes as actual LOC. The manual revision of HTML files reports basically that this is, based on the context, usually ambiguous and both the domains are correct. Overall, the performances can be considered greater with respect to the actual F1-score evaluations.

4.3.3 Multilingual-BERT: Laws De-Identification

The final part of this project wants to actually test the model on de-identification over quotes of laws. The German Legal-NER is used and many different considerations will be presented over each dedicated paragraph. Due to the lack of Italian data, an unsolvable problem is related on making a validation set. Indeed, a German validation set would be useless on detecting good performances over every attempt, since it gives us no information over Italian. That's why only training and test set are used, exploiting previous tests information to bypass this issue. For instance, first four layers are always kept frozen, with a training phase of 3 epochs.

Base test Initial tests start by splitting a small part of Legal NER to create a German test set of LEX (laws), just to confirm again the stability of the model. Furthermore, since the complete dataset contains a high number of different entities, the evaluation will consider two different cases: the first one with only LEX and O tags, and the second one with all the entities. Indeed, solving a binary problem is in general a much more easier and it should correspond with an improvement of scores. Next tables list all the results, reporting in any case just the outcomes related to LEX entity, using the following name format: *m-BERT_LEX_flag-all-entities*.

	precision	recall	f1-score	support
B-LEX	0.95	0.97	0.96	3255
I-LEX	0.96	0.98	0.97	19888
macro avg	0.96	0.98	0.97	23143
weighted avg	0,96	0,98	0,97	23143

Table 4.30: m-BERT_LEX_ALL (German-German)

	precision	recall	f1-score	support
B-LEX	0.94	0.96	0.95	3255
I-LEX	0.95	0.97	0.96	19888
macro avg	0.95	0.97	0.96	23143
weighted avg	0,95	0,97	0,96	23143

Table 4.31: m-BERT_LEX_BINARY (German-German)

Unexpectedly, complete entities model gives better predictions. Analyzing the structure of the dataset, the explanation is that Legal-NER is composed by a lot of legal-linked entities (not only LEX), and classifying these ones uniquely with the O tags creates serious misunderstandings over the context of the sentences. This highlights again the extreme sensitivity of the model to every single variation, which somehow directly suggests the best setting to apply. Following this logic line, every next session test will no more consider the binary case.

German-Italian test Last phase begins to monitor laws prediction using Legal-NER training set over Italian dataset, composed by REDIT data with in addition some sentences from the artificial LEX dataset (already presented in chapter 3), extracting both sentences with or without the LEX entity in the same exact quantity. Moreover, considering the probability of being able to use some Italian data, we exploit the rest of the artificial LEX dataset to improve the training phase. Recent studies, indeed, demonstrate that adding just a small bunch of target language sentences inside the training set improves significantly the overall performances. To have a clear representation of this phenomenon, different tests have been processed, iteratively increasing for every step the total number of Italian phrases. Next tables show with a logic sequence all the obtained results, indicated by the name format *m-BERT_LEX_num-italian-sentences*.

	precision	recall	f1-score	support
B-LEX	0.77	0.49	0.60	1646
I-LEX	0.83	0.58	0.68	11742
macro avg	0.8	0.53	0.64	13388
weighted avg	0,82	0,57	0,67	13388

Table 4.32: m-BERT_LEX_0 (German-Italian)

	precision	recall	f1-score	support
B-LEX	0.81	0.59	0.68	1646
I-LEX	0.81	0.78	0.79	11742
macro avg	0.81	0.69	0.74	13388
weighted avg	0,81	0,76	0,78	13388

Table 4.33: m-BERT_LEX_50 (German-Italian)

	precision	recall	f1-score	support
B-LEX	0.87	0.59	0.70	1646
I-LEX	0.83	0.83	0.83	11742
macro avg	0.85	0.71	0.76	13388
weighted avg	0,83	0,80	0,81	13388

Table 4.34: m-BERT_LEX_100 (German-Italian)

	precision	recall	f1-score	support
B-LEX	0.85	0.85	0.85	1646
I-LEX	0.93	0.87	0.89	11742
macro avg	0.89	0.86	0.87	13388
weighted avg	0,92	0,87	0,89	13388

Table 4.35: m-BERT_LEX_5000 (German-Italian)

The model is incredibly powerful and, starting by non-optimal scores with no Italian LEX data, even with a very low number of examples is able to extract all the needed context to highly improve the overall performances. This is incredibly important, since it confirms that multilingual-BERT could be generally employed (independently from the given entity), with the certainty that evaluations can be boosted just by adding a small package of target language sentences inside the training set.

These tests conclude all the life-cycle of the project, leaving space now to the application of the model over every feasible set of entities.

5 Conclusions and Future Works

This work laid the foundations for a bigger project that will be developed on the next years. The idea is to exploit similar procedures over different new entities, such as religion, political parties, etc.. The first part of the project confirms the fact that, when complete monolingual datasets are available, neural networks models such as Bi-LSTM and BERT can work very efficiently with high evaluations score for each entities. The second part of the project, instead, has analyzed a possible solution to the lack of data to train the them. Exploiting multilingual models and Transfer Learning techniques, knowledge from different languages can be shared to obtain similar evaluation scores of monolingual models over each possible target language. Furthermore, performances can be increased over years, exploiting direct users feedbacks over possible wrong prediction, with a sort of continuous fine-tuning cycle that can keep updated the model and improve its general knowledge.

References

[1] Y. Bengio, P. Simard, and P. Frasconi. “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166. doi: 10.1109/72.279181.

[2] Jason P. C. Chiu and Eric Nichols. *Named Entity Recognition with Bidirectional LSTM-CNNs*. 2015. doi: 10.48550/ARXIV.1511.08308. url: https://arxiv.org/abs/1511.08308.

[3] Ronan Collobert et al. “Natural Language Processing (Almost) from Scratch”. In: *Journal of Machine Learning Research* 12 (Feb. 2011), pp. 2493–2537.

[4] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. doi: 10.48550/ARXIV.1810.04805. url: https://arxiv.org/abs/1810.04805.

[5] Cicero Dos Santos and Bianca Zadrozny. “Learning Character-level Representations for Part-of-Speech Tagging”. In: vol. 5. July 2014.

[6] Joseph L. Fleiss and Jacob Cohen. “The Equivalence of Weighted Kappa and the Intraclass Correlation Coefficient as Measures of Reliability”. In: *Educational and Psychological Measurement* 33.3 (1973), pp. 613–619. doi: 10.1177/001316447303300309. eprint: https://doi.org/10.1177/001316447303300309. url: https://doi.org/10.1177/001316447303300309.

[7] Kristian Nørgaard Jensen, Mike Zhang, and Barbara Plank. “De-identification of Privacy-related Entities in Job Postings”. In: *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa)*. Reykjavik, Iceland (Online): Linköping University Electronic Press, Sweden, May 2021, pp. 210–221. url: https://aclanthology.org/2021.nodalida-main.21.

[8] Xuezhe Ma and Eduard H. Hovy. “End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF”. In: *CoRR* abs/1603.01354 (2016). arXiv: 1603.01354. url: http://arxiv.org/abs/1603.01354.

[9] Tomas Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems*. Ed. by C.J. Burges et al. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

[10] Teresa Paccosi and Alessio Palmero Aprosio. *KIND: an Italian Multi-Domain Dataset for Named Entity Recognition*. 2021. DOI: 10.48550/ARXIV.2112.15099. URL: https://arxiv.org/abs/2112.15099.

[11] Teresa Paccosi and Alessio Palmero Aprosio. *REDIT: A Tool and Dataset for Extraction of Personal Data in Documents of the Public Administration Domain*. 2021. URL: https://ceur-ws.org/Vol-3033/paper58.pdf.

[12] Jeffrey Pennington, Richard Socher, and Christopher Manning. *GloVe: Global Vectors for Word Representation*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162.

[13] Erik F. Tjong Kim Sang and Fien De Meulder. "Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition". In: (2003). DOI: 10.48550/ARXIV.CS/0306050. URL: https://arxiv.org/abs/cs/0306050.

[14] Ashish Vaswani et al. *Attention Is All You Need*. 2017. DOI: 10.48550/ARXIV.1706.03762. URL: https://arxiv.org/abs/1706.03762.

[15] Yonghui Wu et al. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. 2016. DOI: 10.48550/ARXIV.1609.08144. URL: https://arxiv.org/abs/1609.08144.