POLITECNICO DI TORINO
Master degree course in Computer Engineering

Master Degree Thesis

Semantic Annotation of Clinical Notes

Supervisor
Prof. Maurizio Morisio

Candidate
Jeanpierre Francois
Student ID: 243920

Internship Tutor
Dott. Ing. Phd. Giuseppe Rizzo

Academic Year 2019 - 2020

Summary

This work describes how we addressed the real-world challenge of extracting information from the summaries of unstructured and non-standardized Italian electronic medical records related to breast cancer cases. These summaries, also known as GIC-Post records, are compiled by several different specialists who compose an interdisciplinary group of cure (GIC) and are provided by an healthcare facility specialized in breast cancer. The main task is a natural language processing exercise that consists in recognising and tagging sequences that carry crucial information regarding the patient's tumor status which is a Named Entity Recognition (NER) activity. The ultimate goal is to provide to professionals a tool that is able to automatically annotate these clinical records. The realization of this assignment was carried out as a joint partnership between multiple entities belonging to different backgrounds, our contribution lies in the realization of the software infrastructure which consists of a web-based annotation tool and a neural network model for NER.

A class taxonomy document was provided by domain experts that identified as our objective 18 original label classes, these classes do not overlap at all with those normally studied in the literature therefore in order to train a machine learning model we collected a gold standard dataset through manual annotation by domain expert. Starting from the doccano open-source project, we developed an intuitive web application focused on sequential labeling where the manual annotation took place. The quantity of training data examples obtained was very limited whereby, to improve our scores, we added deductive knowledge encoded as rules in the entity recognizing process. The model

we adopted using the spaCy open-source library is a good compromise between performance and costs, its implementation is based on fast a Convolutional Neural Network (CNN).

The solution pipeline schema is illustrated in the following figure:

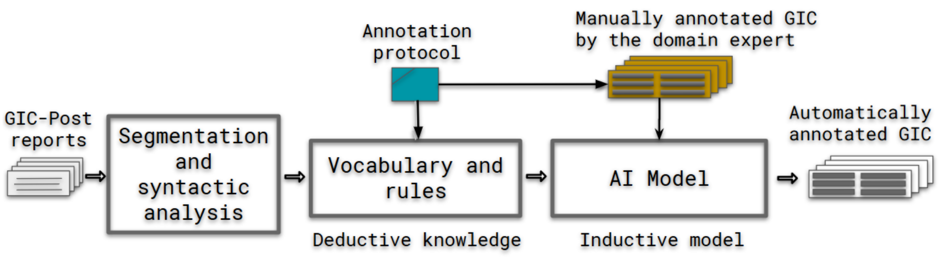

GIC-Post reports
Segmentation and syntactic analysis
Annotation protocol
Vocabulary and rules
Deductive knowledge
Manually annotated GIC by the domain expert
AI Model
Inductive model
Automatically annotated GIC

Figure 1. Methodology solution schema

In our solution we also expanded Italian word vectors with domain specific clinical vocabulary to pre-train the CNN which is used to add in-context information to the embeddings. This hybrid solution composed of a deductive rule-based knowledge paired with a CNN-based inductive model obtained an overall F-Score of 0.56 which, taking into account the shortage of training data, the lack of tools and resources available for non-English languages and completely unconventional class labels, is an interesting result.

Furthermore, using huggingface-transformers repository we experimented the implementation of a way more large and complex BERT-based model for token classification that is currently achieving state-of-the-art results. Once fine-tuned with our domain specific training dataset, this model alone achieved an overall F-Score of 0.61 but required much more computational resources than the hybrid solution.

Abstract

This master thesis work outlines the approach chosen to manage the extraction of information from unstructured and non-standardized Italian electronic health records of breast cancer summaries. This real-world challenge is a Named Entity Recognition activity that faces completely original classes that do not overlap at all with those normally studied in the literature. Taking into account feasibility and speed, we propose a solution that applies natural language processing techniques to fine-tune a statistical model trained onto a narrow set of annotated data. In order to obtain a gold standard we collaborated with domain experts to manually annotate the data collection. A class taxonomy document was provided by domain experts and, to reduce the necessary human effort and speed up the process, we developed a user friendly web-based annotation tool. Furthermore to compare the obtained results, we implemented BERT-based models on the same task.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This master thesis work aims to explain the challenge that we faced and the solution we proposed to solve it. The content of this chapter is a preface to the work and provides an introduction about the field and its macrocosm of applications.

1.1 Digital Data

As years go by, the progress of technology has revolutionized human lifestyles deeply, we are more and more interconnected and the data that we generate is able to travel around the world at light speed. Nowadays we collect a lot of data from a cornucopia of sources and, thanks to the transition from paper to digital, even what was firstly available only on sheet today is now recorded on computers. Collecting and store data in a digital form is of course useful for its reliability, persistence, volume, transience and easier access but is not always easy to analyze, extract useful information and understand a big quantity of data. Collected digital data may be human or machine generated and are divided in two macro types:

- Structured: comprised of clearly defined data types
- Unstructured: everything else

When data is well structured its data types patterns make it simple to systematically search and extract information with performing and solid databases and, when it scales a lot in quantity and complexity, with big data techniques. On the contrary when data is unstructured, indexing and analyses of information result in being a hard problem. Some example of unstructured data are:

- Text files
- Media
- Social Media Data
- IoT Sensors Data

In order to analyse unstructured digital data, tools for specific use cases, based often on machine learning techniques, are being studied every day. In particular for text files, the field of linguistics, computer science, information engineering and artificial intelligence that sets itself the goal of making computers process and analyze natural language data or rather unstructured human generated text is called **Natural Language Processing**.

1.2 Natural Language Processing

The roots of Natural Language Processing (NLP) [1], also known as computational linguistics, trace way back in time down to 1950s, when Alan Turing proposed the Turing Test as a criterion of intelligence. In the early days, NLP systems were based mostly on rule based approaches that focused on hard-coded manually written set of rules. In the late 1980s and mid 1990s happened the so called *statistical revolution* that consisted in applying machine-learning paradigms to automatically learn rules through analysis of large corpora of typical real-case examples that were previously deducted from manual data analysis. Systems based on machine-learning algorithms for automatic learning procedures exploit statistical inference to create models able to generalize, robust to unfamiliar input but requiring a substantial quantity of input data as examples to learn. However, more man-hours are needed to generate more examples, which generally does not increase the complexity of the annotation process. On the other hand, handwritten rules may require fewer examples to deduce, but they are way less flexible and complexity can only increase.

More recently in 2010s, thanks also to the advancement of technology, new models and techniques based on representational learning and deep neural networks emerged as game changing factors in achieving farther state-of-the-art results. Some of the applications of Natural Language Processing are the following:

- Speech
  - Speech Recognition
  - Text-to-speech
- Syntax
  - Lemmatization
  - Stemming
  - Parsing
  - Part-of-speech Tagging
- Discourse
  - Text Summarization
- Semantics
  - Lexical Semantics
  - Machine Translation
  - Natural Language Generation
  - Question Answering
  - Sentiment Analysis
  - **Named Entity Recognition**

1.3 Named Entity Recognition

Named Entity Recognition (NER) [2], also known as entity extraction, identification or chunking, is an information extraction technique that aims to solve the problem of locating and classifying named entity residing in unstructured text data according to a pre-define set of class labels or categories. The goal of this technique is to recognize various named entities residing in a text where a named entity is a *Real-world object* with a name that refers to a pre-defined category such as a person, a country or a company.

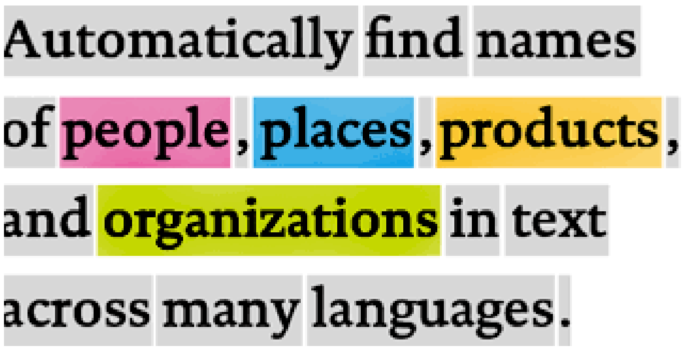


Figure 1.1. Named Entity Recognition [3]

For decades NER has been developed and studied in its aspects with knowledge-based systems, unsupervised systems to feature engineered supervised or semi-supervised learning algorithms. However it is in the latest years that the achievements obtained over NER, as many other NLP sub-tasks, increased greatly thanks to the implementation of deep learning techniques and models. One of the main tasks of this work is a NER activity.

Chapter 2

Deep Neural Networks for NLP

In the following chapter will be given an overview of some of the most recent advancements and techniques used in Named Entity Recognition.

A neural network is composed of three kind of layers:

- Input Layer
- Hidden Layer
- Output Layer

The elementary units that make up a layer are the interconnected nodes, also known as artificial neurons. Each node compute a weighted sum on the input received from other neurons then, after applying a non-linear transformation or rather activation function, fires its output. When the output is finally generated through forward-propagation, corrections to the weights are usually done with a technique called back-propagation. How the neurons are connected, the number of layers and what kind of activation functions is used are the main characteristics of a neural network. Generally networks with a multiple number of hidden layers are called deep neural networks.

2.1 Word Embedding

Computers do not understand words therefore a conversion from the original text is required to achieve a numerical representation. One of the basic idea to obtain a representation is the One-Hot encoding that is a simple method to vectorize categorical variables like words, it generates binary vectors from a text where every element is a zero except the one that represents the word. This simple technique do not acquire any information about semantic or position.

Word embedding is one of the most used techniques that let a computer learn the representation of not only the syntactic but also the semantic meaning of a word and its relationship with other words, every representation is then associated to a fixed size vector. While in One-Hot encoding each word is independent, the idea of word embedding is to generate distributed representations of the words taking into account their dependency between one another therefore words with the same meaning will have a similar representation. The algorithms for word embedding that will be discussed are word2vec, GloVe and FastText.

word2vec

word2vec is set of statistical methods developed by Tomas Mikolov in 2013 [4] that uses shallow neural networks with a softmax layer that selects the most probable word amongst the various possibilities. These models learn information in an unsupervised way about the word in its local context of its neighbouring past and future words within a selected window of predefined size. word2vec uses two different training methods involving a neural network:

- Skip Gram
- Common Bag Of Words (CBOW)

The biggest difference between the two is the way they generate word vectors. The CBOW model is a method that learns embeddings by predicting a word as target and taking its context as input while Skip

Gram predicts the surrounding words given the current word $w(t)$ as input.

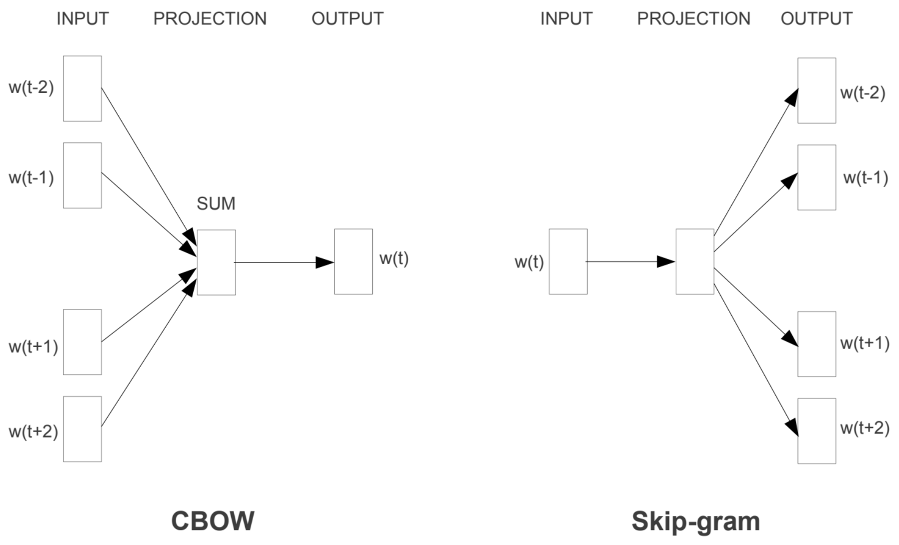


Figure 2.1. word2vec Models [5]

One of the interesting aspects of word2vec multi-dimensional embeddings is that words acquire vector properties that allow geometric operations on word vectors.

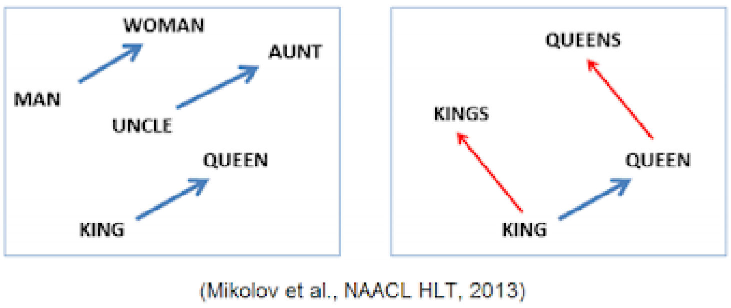


Figure 2.2. E.g. Queens = Kings - Man

GloVe

Global Vectors for Word Representation (GloVe) [6] is an unsupervised learning algorithm that uses aggregated global word-word co-occurrence statistics from a corpus. While word2vec relies on a window to use local information, GloVe aims to capture global and local statistics across the text corpus to generate word vectors. The key idea is that a semantic meaning can be derived from the co-occurrence matrix between words.

FastText

Published in 2016 [7][8], is a revisited extension of the word2vec model that breaks word into sub-words or rather n-grams instead of passing the individual word to the input layer. A vector of words is then built with these sub-words. This approach generates a better representation of rare words and for out of vocabulary words by building a vector representation from its n-grams even if they do not appear in the train corpus. The trade-off of this model is in technical requirements, its memory consumption grows with the number of n-grams and it usually takes way longer to train in comparison to word2vec and GloVe.

Beyond Word Vectors

The generated word vectors obtained with the algorithms listed above provide for each word in their vocabulary a vector that represents the meaning of the word. These algorithms share the important assumption that the meaning of a word is relatively stable between sentences, or rather that their meaning does not change based on the context. [9] This assumption is in fact a limitation due to the polysemic nature of words. For example the word *bank* in the sentences *the bank is a well known credit institution* and *The bank of the river is expanding this season* has completely different meanings but it will be represented with the same vector. Consequently a word in general should have different vector representations for each of its meanings, to achieve this with these embeddings a solution is to characterize the obtained embedding with information of the surrounding word embeddings while analyzing a text corpus without changing it completely as showed at Chapter 4.

Softmax

The SoftMax Function [10], also known as softargmax or normalized exponential function, is used to normalize and convert an input vector of real numbers into a probability distribution proportional to the exponentials of the input. As in word2vec, this function is heavily used for classification purposes in neural networks to map their output to distributions over predicted class labels.

$$\sigma : R^{\kappa} \rightarrow R^{\kappa}$$

$$\sigma(z)_i = \frac{\exp z_i}{\sum_{j=1}^{\kappa} \exp z_j} \qquad for \quad i = 1, ..., \kappa \qquad z = (z_1, ..., z_{\kappa}) \in R^{\kappa}$$

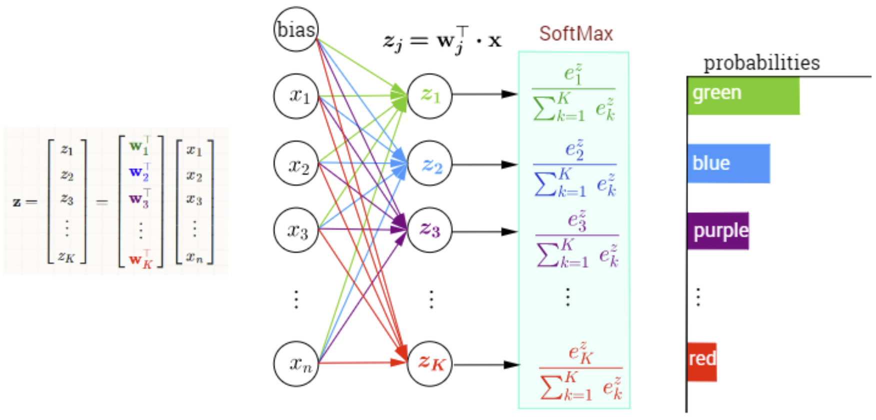

$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_K \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^\top \\ \mathbf{w}_2^\top \\ \mathbf{w}_3^\top \\ \vdots \\ \mathbf{w}_K^\top \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$
bias
x_1
x_2
x_3
x_n
$z_j = \mathbf{w}_j^\top \cdot \mathbf{x}$
z_1
z_2
z_3
z_K
SoftMax
$\frac{e_1^z}{\sum_{k=1}^K e_k^z}$
$\frac{e_2^z}{\sum_{k=1}^K e_k^z}$
$\frac{e_3^z}{\sum_{k=1}^K e_k^z}$
$\frac{e_K^z}{\sum_{k=1}^K e_k^z}$
probabilities
green
blue
purple
red

Figure 2.3. Multi-Class Classification with a neural network and SoftMax Function [11]

2.2 Recurrent Neural Networks

During the analysis of an unstructured text is usually useful to carry information related to the order of words or sentences, to what has already been read. Recurrent Neural Networks (RNN) maintain a memory about the previous elements while processing new ones by sharing information of consecutive inputs belonging to the same sentence. To learn the meaning of a word it may be necessary to look at words ahead in the sentence thus to catch not only forward dependencies but also backward ones it can be used two RNN, one for each direction, forming the so called bidirectional RNN.

In RNNs single neurons share information back to themselves to maintain a memory but it dilutes as iterations go by forgetting the distant past. However in some cases it is useful to remember information in the far past whereby to solve this issue a complex design of RNNs were invented that uses several neurons connected in a manner focused in retaining, forgetting or disclosing peculiar information. These complex networks, frequently used in blocks, are called Long Short-Term Memory (LSTM).

Another important simpler but with similar performance implementation of LSTM is called Gated Recurrent Unit (GRU).

2.3 Transformers

One of the milestones of Machine Translation in recent years is undoubtedly Attention which influenced also several other NLP subtasks such as NER. As humans focus on certain portions while watching an image or some sentences while reading a text, attention let networks focus on important information residing in the input. Introduced for sequence-to-sequence learning by Bahdanau in 2015 [12], attention in the proposed solution allows a decoder to exploit information about the source sequence RNN's hidden states, that are handed over to the decoder as a weighted average as additional input.

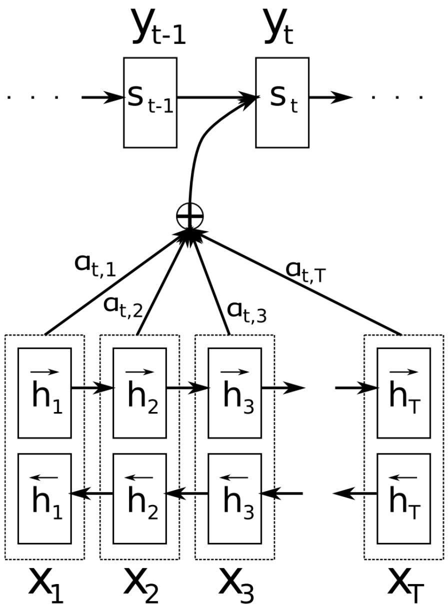


Figure 2.4. Graphical illustration of the model proposed by Bahdanau et al. trying to generate the t^{th} target word y_t given a source sentence $(x_1, x_2, ..., x_T)$ [12]

Amongst the implementations that go back to this technique there are convolutional, intra-temporal, gated and self-attention. In self-attention the focus is placed upon the neighbouring words in a sentence in order to obtain a contextually sensitive word representation. A model based solely on attention mechanism is the Transformer [13], its architecture is composed of multiple self-attention layers stacked together to catch global input-output dependencies and it is free of RNNs and convolutions. This model is composed of multiple encoders and decoders stacked on top of each other, where each of them contains a self-attention mechanism, and cross-attention between encoders and decoders. To compute attention, which can be described as a mapping between a query and a set of key-value pairs to an output, three vectors are created from encoder's input vectors:

- Query Vector
- Key Vector
- Value Vector

For each word the attention is calculated by taking the dot product of its query vector with the key vectors of all the words in the input and then scaling the result dividing by the square root of the key vector dimension (d_k) whereupon applying a softmax function. The sum of the softmax results multiplied by the value vector is the self-attention vector of the word. The attention computed on a set of queries Q with the corresponding key-value matrices K and V results in a matrix of outputs given by the following formula:

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

In the Transformer's architecture, attention is not computed in a single attention function but it is calculated multiple times at different positions, in parallel and independently. Their different representation sub-spaces are then merged together by the multi-head attention module through concatenation and linear transformation.

To exploit positional information without recurrence and convolutions, the transformer model adds positional encodings to the input embeddings of an equal dimension d_{model}. Position-dependent sinusoidal functions are used in the original paper:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{100000^{2i/d_{model}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{100000^{2i/d_{model}}}\right)$$

Where pos is the position and i the dimension.

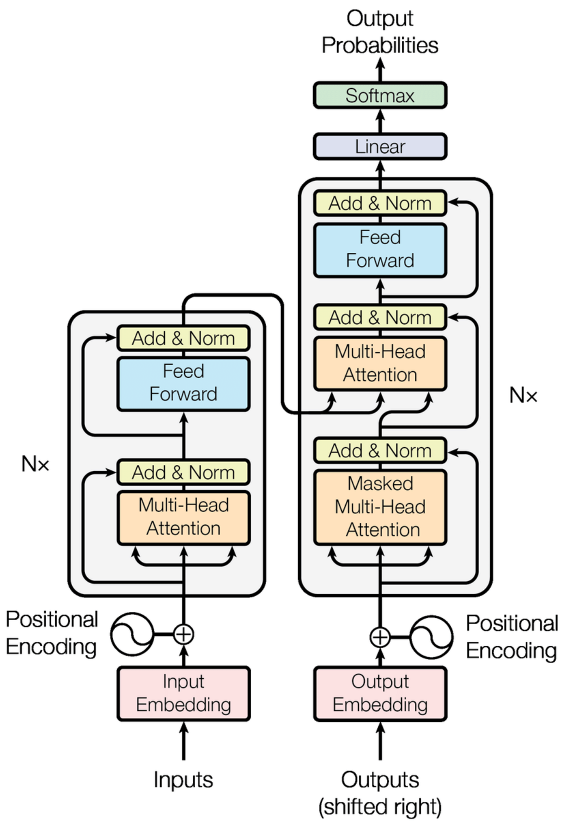


Figure 2.5. The Transformer - model architecture [13]

2.4 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a kind of artificial neural feed-forward network inspired by animal visual cortex. Using convolutions as primary operation which is a basic graphic operation, they are way faster than other networks such as RNNs or Transformers specially if implemented on GPUs. In NLP, CNNs are often used for classification where its input of this networks are usually word embedding vectors while its output is given to a softmax layer.

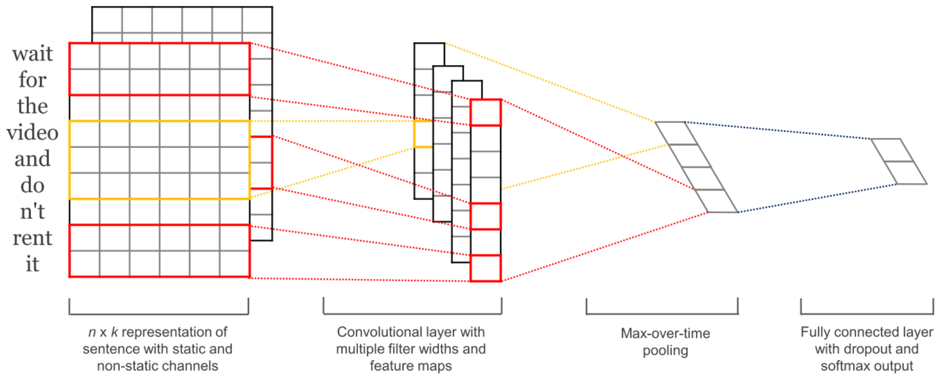

wait
for
the
video
and
do
n't
rent
it
n x *k* representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 2.6. An example of CNN for Sentence Classification[14]

CNNs can also be used to enrich a word embedding with information about its actual context as showed further at Chapter 4.

Chapter 3

Problem Definition

The purpose of this master thesis project is to provide an accurate report of the solution designed to create a series of tools aimed to automate the extraction of information from clinical documents and facilitate its reading.

This work has used Italian interdisciplinary Electronic Health Records of breast cancer cases, but its methodology can be applied to any similar scenario. This chapter will provide a better understanding about how interdisciplinary medical reports used in this work are produced within the oncological network of the Italian regions of Piedmont and Aosta valley and our goal regarding their use.

3.1 Electronic Health Record

The patient care process starts from collecting updated knowledge about his health state when he is taken in charge. Patient information such as his clinical history, what test he underwent and so on in many countries are stored in a digital record. An electronic health record traces patients clinical history but often contains also all of the administrative information pertinent to the cure process done under a certain entity such as an hospital or a house of cure. In Italy this kind of records are also known as *Fascicolo Sanitario Elettronico* (FSE). This collection of information usually contains, amongst other things, patient registry, status of the cure, complications that occurred, previous clinical history, laboratory analysis outcomes and medical visit reports.

The data set under consideration is made of a specific kind of EHR produced by several different specialists who compose a so-called *Group of Cure*.

3.2 Interdisciplinary Group of Cure

In October 2006, the European Parliament adopted a resolution to urge Member States to ensure the presence of multidisciplinary breast cancer centers in all areas of the country.

In Italy these facilities, grouped into sub nets by region, are called *Breast Units*. Each one treats more than 150 cases every year and must have at least a core team of six dedicated professionals:

- Radiologist
- Surgeon
- Pathologist
- Oncologist
- Radiotherapist
- Data manager

This group of experts is named interdisciplinary group of cure (GIC).

When a patient is diagnosed with a breast tumor her clinical notes and analyses are gathered together and then a GIC meeting is scheduled to decide how to proceed. During these meetings an overview of the patient clinical status is compiled into a GIC report that in cases where surgery was needed may be PRE-surgery or POST-surgery, the latter are called GIC-Post reports. An important aspect about these reports is that they are written in Italian as unstructured text.

Our data is made of GIC-Post surgery reports provided by a breast unit, each one of them contains patient's clinical history referred to the breast tumor and very valuable summary data.

3.3 Our Goal

GIC reports, as many electronic health records, are written in an unstructured and non-standardized format on electronic healthcare systems. Prior to this work, a method to inspect relevant information from those documents was not available. In order to exploit those useful data not yet used as much as they could due to the manual labor required, we have oriented ourselves towards the automation of the analysis of these texts.

Our objective consisted in creating an automatic pipeline for analysis, recognition and extraction of valuable information stored inside GIC-Post reports utilizing Natural Language Processing (NLP) techniques. NLP techniques are able to determine the meaning of the information that lies in unstructured data and in doing so can highly reduce human effort. The indicators and key information that we are looking for inside these electronic health records are single words or sentences that represent entities with a clinical useful meaning for professionals. Our goal is to identify each one of those entities and couple them to a specific predefined label class. This activity is also known as **Named Entity Recognition**.

Chapter 4

Approach

This chapter describes the methodological idea behind the proposed solution pipeline, starting with the recovery of the gold standard data set needed to train models capable of extracting valuable information from clinical records.

4.1 Main Idea

Our plan is to design a tool that lets doctors and professionals access crucial knowledge stored in medical reports by creating a user friendly platform coupled with a natural language processing model able to recognise key information in unstructured text data.

In an attempt to implement this idea while maintaining a good compromise between performance and usability, we decided to implement a hybrid solution using a lightweight computational model, which adapts to our web-based environment, formed by a neural network integrated with a rule-based approach. To generate examples for training these models, we created a custom web-based annotation platform where a domain expert (e.g. a surgeon) manually annotated reports.

4.2 Methodology

One of the biggest challenge of Natural Language Processing applied to real world production domains is the lack of labeled data sets.

Generally we would try to exploit transfer learning for NLP through implementation of models trained on very large data sets and fine tuned onto smaller collection of data related to a specific case of study but our problem in particular aims to answer a named entity recognition task on a very peculiar and technical field with unique classes and vocabulary. In fact, the classes we want to predict do not overlap at all with those normally studied in the literature, therefore we are unable to use any of the publicly available data sets commonly employed for this activity. To remedy the absence of a data set that matches our specific needs, we had to create one from scratch.

Our solution implements a multi-step process that we can group into two main phases. The initial phase includes a customization and distribution of a web-based annotation tool suited around the needs of our selected experts, followed by the definition of a class taxonomy document. After this configuration, the manual annotation phase begins, after which the manually obtained gold standard is used to train a neural network in a supervised way in an effort to obtain a model able to automatically annotate entities residing in GIC-Post documents.

4.2.1 Taxonomy of Classes

Labeled data of very high quality with annotations as close as possible to the ground truth is usually referred in literature as gold standard.

In order to conduct a supervised training and evaluation of models, gold standard corpora for reference are vital. To obtain a gold standard, a manual annotation step often is needed. Manual annotation is the act of adding further knowledge to the data (meta-data) that will allow information extraction. This activity is usually undertaken by humans and despite being more labour intensive than automatic annotation it is considered to be more precise.

In domains like healthcare, a domain expert is usually required to recognise and determine which entities effectively contains useful meaning. A class taxonomy document was provided by domain experts in which a list of characteristics and key indicators to be searched in the reports is defined. This document achieved a complete characterization of the named entity recognition problem and the collection of 18 labels to work with:

- **Stadio**: identifies the stage of the breast cancer following AJCC 2017 standard classification (eighth edition) [15]
- **Neoadiuvante**: neoadjuvant which is a synonym for chemotherapy
- **Visita oncogeriatrica**: oncogeriatric visit
- **RM Mam**: patient's breast magnetic resonance imaging
- **Biopsia**: patient's biopsy
- **T**: tumour stage classification following AJCC 2017 standard [15]
- **N**: regional lymph nodes classification
- **M**: metastasis detected at distance
- **Micrometastasi**: micrometastases
- **Macrometastasi**: macrometastases
- **ER**: estrogen levels
- **PGR**: progesterone levels
- **Ki67**: antigen KI-67
- **HER 2**: receptor tyrosine-protein kinase erbB-2
- **Invasione vascolare**: indicator of tumor aggression levels
- **Margine**: growth margin

- **Grado**: nuclear grade
- **BRCA**: BRCA test results, if the patient underwent it

Whereupon we instructed domain expert manual annotators about how the annotation activity should be done over our web annotation tool.

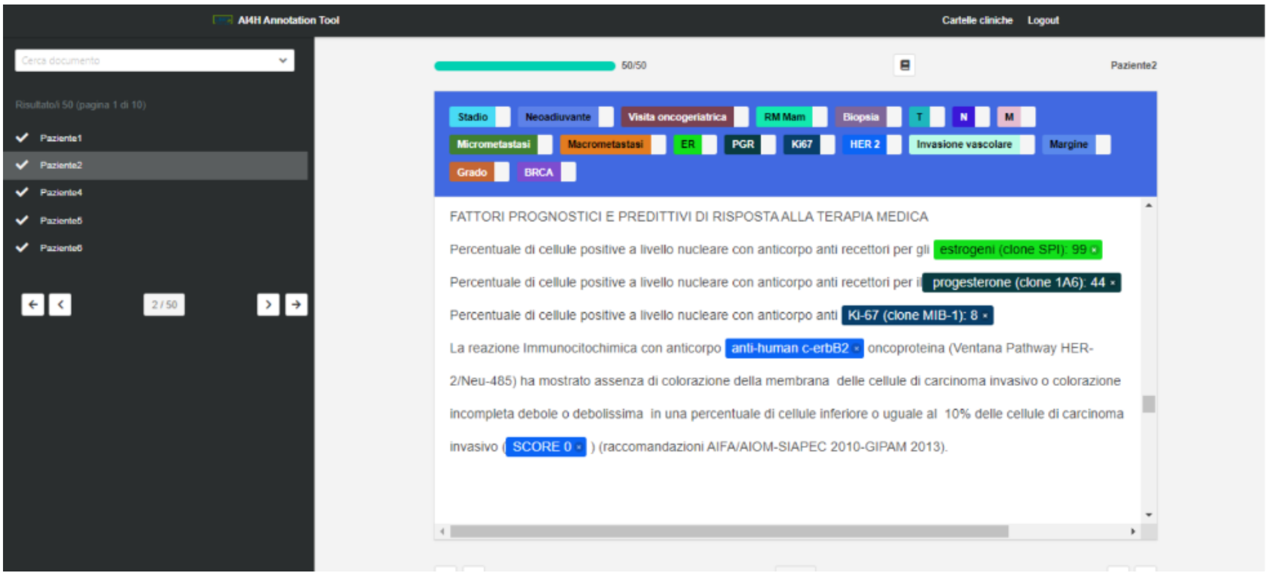
4.2.2 Web-based Annotation Tool

Gold standard data may be hard to obtain due to it being human effort expensive. To reduce the necessary struggle for our domain experts, we prepared an highly user friendly customized web-based annotation tool.

Amongst many available tools specialized in named entity recognition activity, by taking into account the following parameters:

- Complexity
- Customization
- Usability
- Costs

We selected as initial basis doccano [16] that is a modern looking open-source annotation tool project designed for humans meaning that it has a low entry barrier for new annotators. We developed a web application version focused on the named entity recognition activity optimized for efficient, intuitive and quick annotation. Written mostly in Python, Vue and NodeJs it has been deployed as a docker container running on a remote server. This deployment approach lets healthcare facilities to painless host this annotation tool solution on-site using the docker environment.



FATTORI PROGNOSTICI E PREDITTIVI DI RISPOSTA ALLA TERAPIA MEDICA

Percentuale di cellule positive a livello nucleare con anticorpo anti recettori per gli estrogeni (clone SPI): 99

Percentuale di cellule positive a livello nucleare con anticorpo anti recettori per il progesterone (clone 1A6): 44

Percentuale di cellule positive a livello nucleare con anticorpo anti Ki-67 (clone MIB-1): 8

La reazione Immunocitochimica con anticorpo anti-human c-erbB2 oncoproteina (Ventana Pathway HER-2/Neu-485) ha mostrato assenza di colorazione della membrana delle cellule di carcinoma invasivo o colorazione incompleta debole o debolissima in una percentuale di cellule inferiore o uguale al 10% delle cellule di carcinoma invasivo (SCORE 0) (raccomandazioni AIFA/AIOM-SIAPEC 2010-GIPAM 2013).

Figure 4.1. Web-based annotation tool sample

4.2.3 Supervised Hybrid Model

Once we obtained the gold standard data set we were able to start the model training phase. In order to generalize based on the context and beyond specific examples, statistical approaches are the main choices but they become usually weaker in application specific environment with few training data. In these peculiar cases, rule-based knowledge is particularly useful when it can exploit statistical predictions as entity labels, syntactic dependencies, or part-of-speech tags. We combine deductive knowledge, rule-based, with an inductive model.

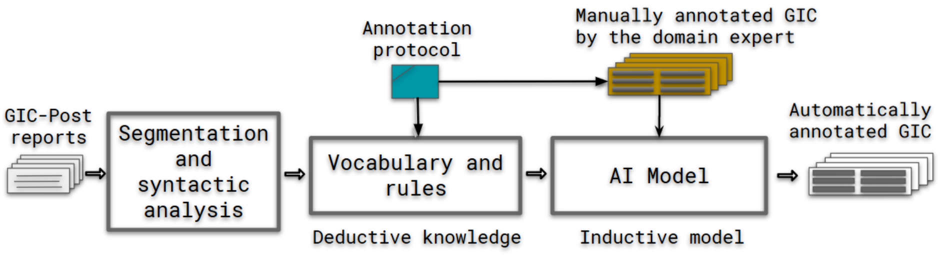
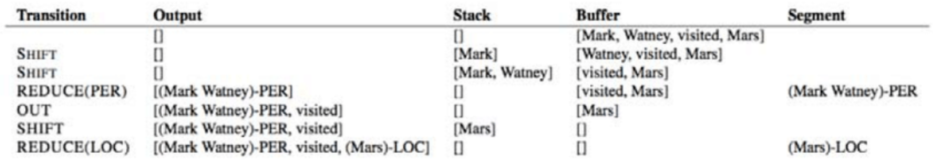

Annotation protocol
Manually annotated GIC by the domain expert
GIC-Post reports
Segmentation and syntactic analysis
Vocabulary and rules
AI Model
Automatically annotated GIC
Deductive knowledge
Inductive model

Figure 4.2. Methodology solution schema

The core of our inductive model is a convolutional neural networks with residual connections whose outputs combined with other status information are forwarded to a linear neural network for label classification. To develop the solution that we propose, which is a hybrid approach implementing a rule-based knowledge and a machine learning model, we relied on the spaCy [17] open-source Python library. Deepening the topic, the inductive model utilizes a transition-based approach for named entity recognition that, instead of focusing on each word as object of interest, exploits a finite state machine methodology. The basic idea is to maintain a state while reading a sequence and make predictions on the actions to be performed. Starting with an empty stack with every word into the buffer and without entities assigned, a list of actions or transitions that change the state is defined and the goal is to predict the sequence of those actions. In our case of study an action corresponds in assigning a word to an entity.



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

Figure 4.3. Transition sequence with a stack-based model example [18]

The statistical model trained to foresee these transitions could be broken down in four macro phases:

- Embed
- Encode
- Attend
- Predict

Embed

The embed phase consist in text tokenization or rather breaking up text into words and subsequently obtain their vector representation. From the Wikipedia's definition of tokenization [19]:

In computer science, lexical analysis, lexing or tokenization is the process of converting a sequence of characters (such as in a computer program or web page) into a sequence of tokens (strings with an assigned and thus identified meaning).

Moreover

Tokenization is the process of demarcating and possibly classifying sections of a string of input characters.

Ordinarily in an embedding table, each word is mapped to a distinct identification number in a vocabulary where each one of the entries store a vector. These look-up tables to store word vectors become quite large in size as the entries in the vocabulary grow in number.

In order to reduce the size of the embedding table, a probabilistic data structure is used based on the hash embeddings [20] idea: alternately to store for each token an embedding vector, their representing vector is built by hashing multiple times from a shared pool of embedding vectors.[1] In details, to generate a unique vector representation of a word and avoid collisions the model extrapolates its lexical sub-features and hashes them four times with different random seeds. The lexical sub-features used are prefix, suffix, norm and shape where norm represents the lowercase version of the word while shape derives from its capital and lowercase letters. The vector representation for each feature is the sum of four different entries in the table meaning that collisions on every sub-features is very unlikely therefore almost all of the words in the vocabulary are going to end with distinct representation with a trade-off of a slower learning process. The obtained separate embeddings for each word sub-features are concatenated together and fed to a simple neural network composed of 1 hidden layer that is a Maxout unit with a layer normalization, this unit's output layer of a predefined width releases a vector that corresponds to the final embedding.

[1]To hash means using a hash function that is a mathematical algorithm that unidirectionally maps data of variable size to an output of fixed size, it usually is written to accept a seed within its input that is a number used to initialize the algorithm to a certain state. An hash table is a data structure mapping keys to values that is accessed by key using a hash function to compute its index.

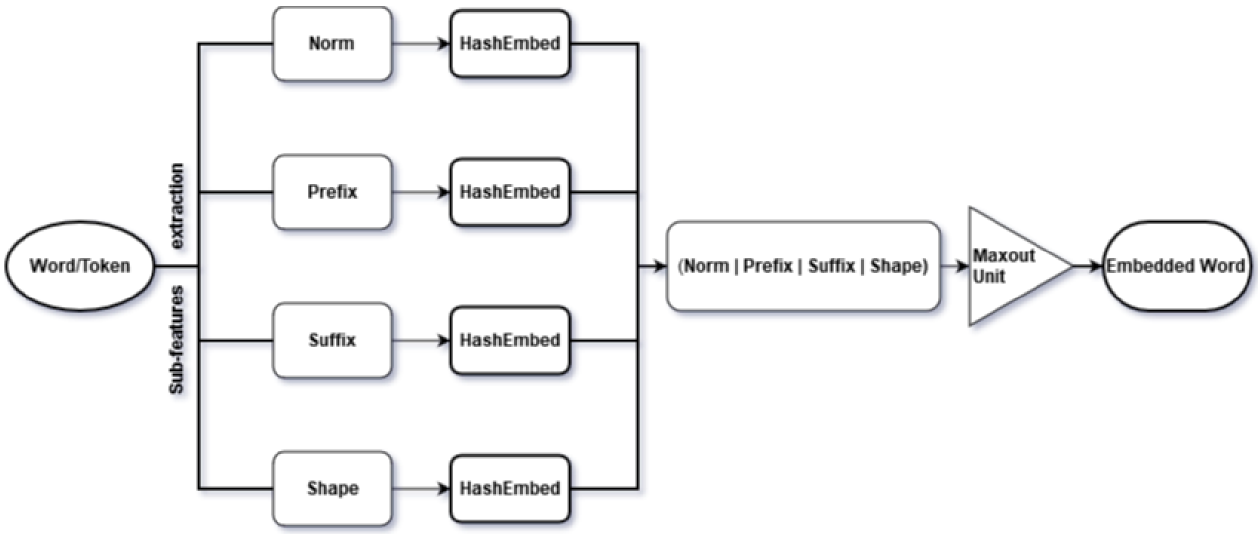


Figure 4.4. Sub-features hash embedding

In the Maxout layer each neuron is broken down in sub-neurons also called pieces, each of which has their own weights and biases. The input to a neuron is forwarded to its sub-pieces that outputs the computations without applying the activation function. The final output of the neuron is the max of the output of all its sub-pieces. Formally for the neuron j of layer i having n pieces:

$$a_j^i = \max s_{jk}^i \qquad \forall k \in [1, n]$$

Where

$$s_{jk}^i = a^{i-1} \cdot w_{jk}^i + b_{jk}^i$$

w_{jk}^i = vector for sub-neuron k of neuron j that contains a weight for every neuron in the previous layer $i - 1$ b_{jk}^i = bias for sub-neuron k of neuron j+

Normalization techniques, like batch normalization, are methods used to boost speed, stability and performance of a neural network. Batch normalization, published in 2015 [21], is used to normalize the input from the previous layer by scaling and adjusting its activation function output. Batch normalization computes the mean and variance of each mini-batch and normalizes each feature according to the mini-batch statistics therefore their mean and variance will differ for each mini-batch.

Layer normalization [22] is a method designed to overcome the drawbacks of batch normalization where the statistics are computed across the batch and are the same for each example in the batch. Based on the notion that one layer affects the summed inputs directed to the subsequent layer with highly correlated changes, it tries to reduce this *covariate shift* effect by computing the statistics of mean and variance across each feature, over all the hidden units in the same layer, independently of other examples. Considering the l^{th} hidden layer and let a_i^l be the vector representation of the summed inputs to the i^{th} hidden unit in that layer.

$$\mu^l = 1/H \sum_{i=1}^{H} a_i^l \qquad \sigma^l = \sqrt{1/H \sum_{i=1}^{H} (a_i^l - \mu^l)^2}$$

Where H is the number of hidden units in a layer. μ refers to the mean while σ to the variance.

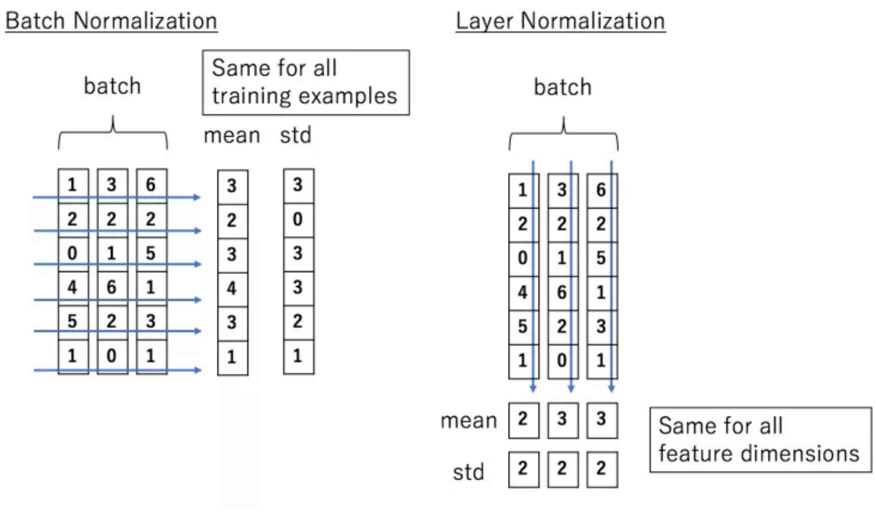


Figure 4.5. Batch and Layer Normalization [23]

Encode

Right after the embedding phase we have at our disposal a sequence of word vectors representing each word individually out-of-context, the encode step goal is to enrich each representation with additional in-context information taking into account neighboring vectors of neighboring words. In order to do so, a trigram convolutional neural network of 4 hidden layers with residual connections is applied for the intention of recalculating the vectors based on the context. The main idea behind this implementation is actually the Collobert and Weston [24] window approach concept that is extracting on both sides of the word a window of words. In details, working within a window it does a convolution with neighboring words that will be fed forward to a Maxout layer. By adding more layers it makes the resulting vector sensitive to subsequent neighboring words on both sides. The residual connections make the output of each convolution layer a mix of its output and input in order to let the network learn a bias while maintaining a word representation similar to the starting one.

Tokenization, embedding and encoding can be wrapped up together and named token-to-vector model or *Tok2vec*.

Attend

The previously obtained in-context word representations are used to generate context-sensitive tensors of embeddings where every row is related to a vector of a word in the text corpus.[2] Those tensors are part of the mechanism generating the weights of the state for the machine within the **attend** phase. While reading linearly the document, the state is computed taking into account immediate neighboring words and previous predicted entities, even those very distant in the text corpus.

Predict

Finally, to predict actions it initially computes the tensors and initialize the weights by multiplying tensors by the first hidden layer. Subsequently, word by word the action to be taken is decided by a multi-layer perceptron to get action probabilities and then choosing given the state the highest probable valid action. In our named entity recognition case the actions, or transitions, to predict refer to BILOU entity tags to be associated to tokens, namely:

- B - beginning of a multi-token entity
- I - token inside of a multi-token entity
- L - last token of a multi-token entity
- O - token outside of named entities

[2]A Tensor is a kind of data structure used in linear algebra, a multi-dimensional array derived from a generalization of vectors and matrices. Most of the computing operations that works on scalars, matrices and vectors can be applied on tensors.

- U - token is a unit-length chunk of an entity

The loss function used for training is the multi-label logarithmic loss, also known as log loss. Loss functions, also known as error or cost functions, measure how well a model is performing a task during the optimization process by mapping a set of parameter values onto a scalar value. In context as machine learning the log loss function, by computing error rates between 0 and 1, becomes the same as the cross-entropy. Its mathematical formula for C label classes (C>2) is the following:

$$L = -\sum_{i=1}^{C} y_{o,i} \log(p_{o,i})$$

Where $y_{o,i}$ is a binary indicator (0 or 1) from the ground truth of whether class label i is the correct classification for observation o. While $p_{o,i}$ is the predicted score for each class that observation o belongs to class i.

Another important loss function is the Least square errors (L2) that is used to minimize the error, which is the sum of the all the squared differences between the true value and the predicted value.

$$L = \sum_{i=0}^{n} (y_i - h(x_i))^2$$

Rule-based and Statistical

The rules that we compiled were built after the manual annotation phase when we had a complete overview of the annotations and their grammatical structure. With a consolidated class taxonomy document, we had the possibility to write down regular expression rules capable of recognising the least variant entities. In order to join knowledge obtained through context specific deductive analysis with the statistical model we placed our rules in the pipeline right before the inductive model. In doing so, named entities selected through rule-based decisions take precedence over those decided by the inductive model.

Language Model pre-training

Having to work with a domain specific very technical language, many clinical words do not have a vector representation in commonly used word vector models, especially for Italian. To solve this issue in order to improve our scores in recognising these special notations, we extended word vectors models with our custom word vocabulary and then trained the token-to-vector module, containing the convolutional neural network, on unlabeled text data. The goal is to fine-tune the network in creating correct in-context word vector representations of our technical vocabulary for GIC report documents. To achieve this task, a BERT-like idea also known as *Language Modelling with Approximate Outputs* [25] is implemented that consists in predicting token's word vector instead of token's identifier. Unlike the common language model, for each word with this technique the model learns to predict a single point in the vector-space to achieve a better token representation that is one vector instead of predicting a probability distribution over the words in the vocabulary. The training is carried out by chaining an output layer on top of the token-to-vector input model and training it as a masked language model with a L2 loss function in order to predict arrays where each row represents a token. The output layer consists in a Maxout unit chained to a fully connected layer ultimately layer normalized.

In Masked language modeling the output is reassembled from partial input, one or more of words in a sentence are masked and then the model has as objective to predict masked words using their context or rather their neighbouring words.

4.2.4 Supervised BERT Model

As a baseline, we wanted to generate results with BERT [26] based models that are one of the currently state of the art solutions.

Bidirectional Encoder Representations from Transformers (BERT) models, by jointly conditioning on both left and right context in all layers, are designed to pre-train deep bidirectional representations from unlabeled text. Many standard language models previous to BERT were unidirectional architectures where every token can only attend to the subsequent one in a single direction (left-to-right or right-to-left) which is sub-optimal especially for token-level tasks such as named entity recognition. In order to relax the unidirectional constraint, BERT utilizes a masked language model pre-training objective where, by randomly masking some of the tokens in the input, the objective is to predict, based solely on its context, the original vocabulary id of the masked word. To learn relation between sentences during pre-train, it also uses Next Sentence Prediction (NSP) where given as input two segments of words the objective is to predict if the latter one is the *next sentence* of the former. The pre-train of these models is run on huge unlabeled data corpus from multiple entities like Wikipedia and BookCorpus, then the expensive training phase that needs very powerful resources and many days takes place.

The obtained pre-trained BERT model can be fine-tuned at a significantly lower computational cost by adding one additional output layer on a specific task like the one analyzed in this work. To preprocess the input to BERT is used WordPiece tokenization [27], each token of every sentence is always a special classification token ([CLS]). Sentence pairs are often packed together and to distinguish between them a special token is used to as a separator ([SEP]). The multilingual model, used also for Italian, comes with a vocabulary of 110k tokens but when it faces an original token, like the ones in specific clinical data, the WordPiece tokenization can enact one of two strategies:

- One-to-one tokenization: the token is in the vocabulary
- One-to-many tokenization: the token is not in the vocabulary

therefore WordPiece will split it into a sequence of vocabulary items of two tokens using greedy longest-match approach, the first one will end with "##" while the latter one will begin with "##".

Consequently the vocabulary may be divided in single or beginning tokens (no leading "##") and ending tokens (with a leading "##").

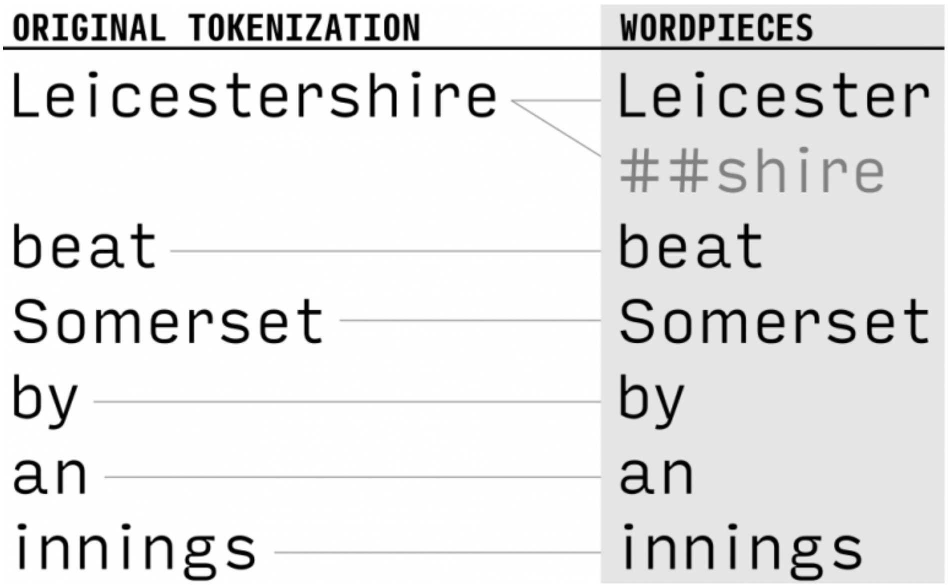


Figure 4.6. WordPiece tokenization example [27]

WordPiece embeddings are one of three input part to the BERT model that summed together with the other two forms the final embedding. The other two embeddings are Segment or Sentence embeddings, which is not significant for named entity recognition and the same segment (e.g. Sentence "A") can be used for every token, and Position embeddings related to the position of the token inside the sequence.

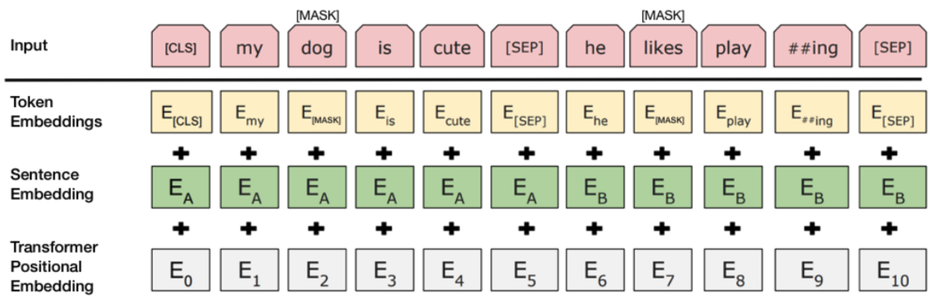


Figure 4.7. BERT input format [26]

During BERT fine-tuning, the pre-trained transformer serves as an encoder and is added on top of it a randomly initialized classifier. For named entity recognition, the classifier is a projection from the token hidden state size to the size of the tag set that then passes through a SoftMax to turn scores into likelihoods. All token positions share the very same classifier.

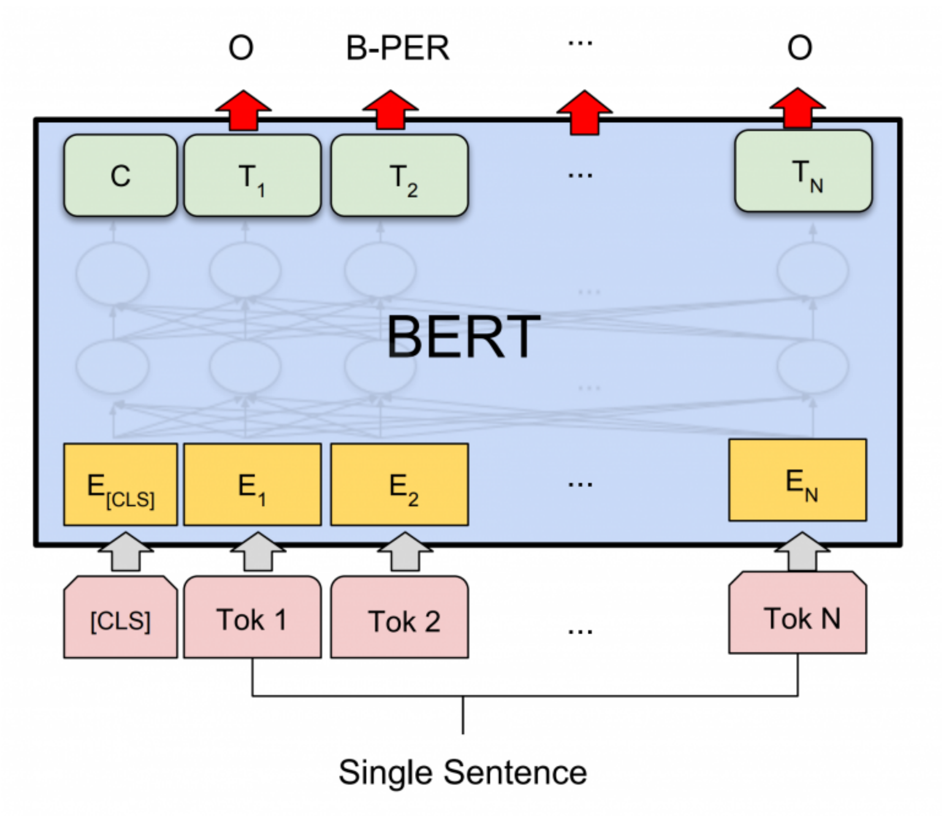


Figure 4.8. BERT for NER [26]

For one-to-many tokenization where an out-of-vocabulary token is split into multiple WordPieces, each WordPiece is tagged by the classifier but only predictions related to single and beginning tokens are included in the loss and in the output at run-time. Therefore beginning tokens represent the entire original token, so the hidden states relating to single and beginning tokens are the only ones relevant in the last layer.

The input for training the BERT classifier follows the BIO entity tagging format:

- B - beginning of a multi-token entity

- I - token inside of a multi-token entity
- O - token outside of named entities

Chapter 5

Experimental Setup

This chapter describes experimental setup details of tools, hardware and software, and the data collection that we used to bring this project to life.

5.1 Dataset

One of the most troubling issue of applying Natural Language Processing techniques is the lack of labeled data. In order to accomplish the goal, this work needed a substantial amount of labeled data to obtain a relevant generalized solution to the problem. Being a highly domain specific task with completely original labels, that do not overlap at all with those normally studied in literature, we could not use any of the available public data set. To remedy to this absence of data we created a data collection on our own and we hope that with the obtained automatic annotation tool it can grow a lot more for further experiments and fine-tuning.

The data set available is formed of a collection of GIC post surgery summary reports. Inside each report, many words are related to a domain specific healthcare vocabulary. Through a joint collaboration with an healthcare facility we retrieved 50 GIC post summaries of breast cancer surgeries, each one is a very long text that contains a variable number of words and lines with an average of 1427.32 words per documents. The total data corpus counts 71366 words.

Once uploaded onto our web-based annotation tool application, a manual annotator created a gold standard using the 18 labels described previously in the class taxonomy document in order to highlight and add meta information to the corpus. The resulting gold standard is composed of 730 manually obtained annotations.

GIC-Post surgery summaries analyzed	50
Total data corpus word count	71366 words
Average word count per document	1427 words
Samples of manually annotated entities	730
Average annotation length	2.42 words

Table 5.1. Dataset Statistics

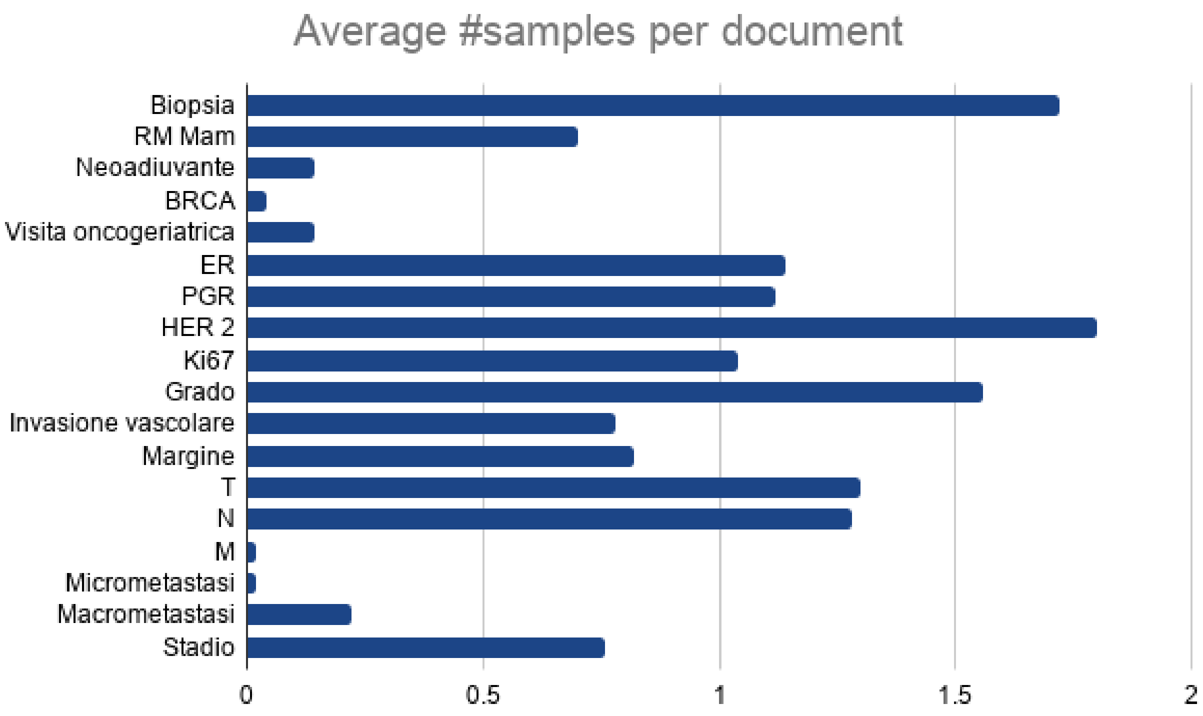


Figure 5.1. Average samples per document

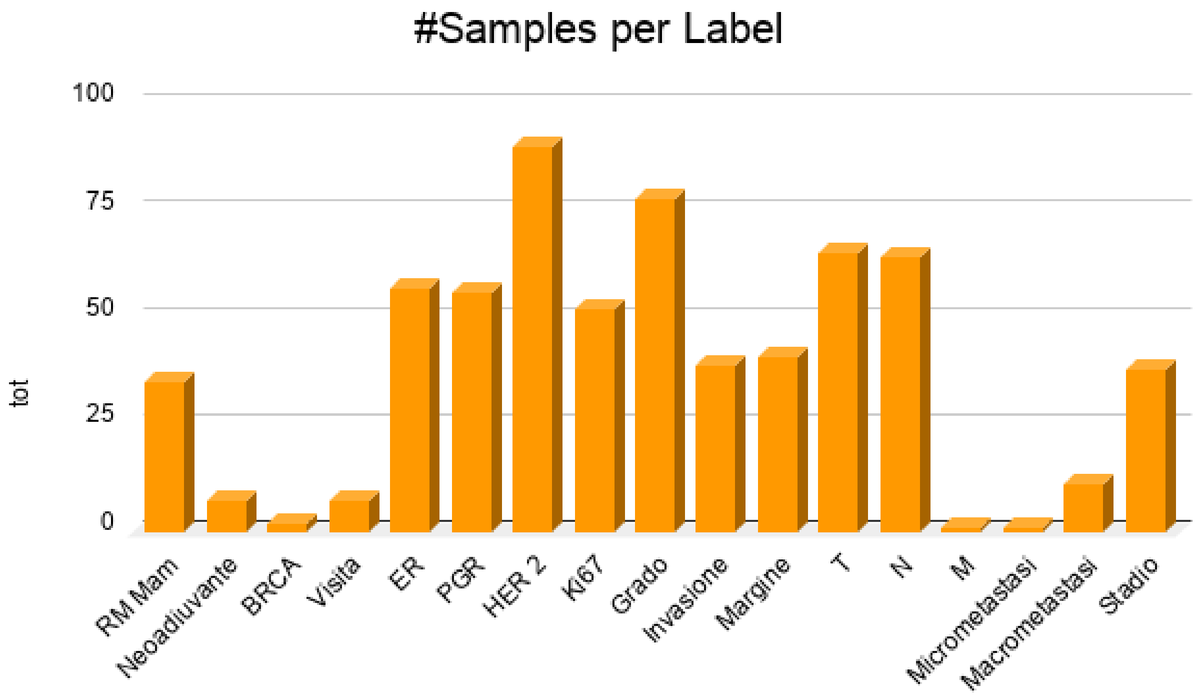


Figure 5.2. Samples per label

5.2 Hardware

Computational resources and hosting services for the web-based annotation tool are provided by Links Foundation. We used a machine whose hardware details are described in the table 5.2 below.

Architecture	x86_64
Service Network	Ethernet Controller 10 Gb/s
CPU Model	Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
CPU(s)	8
Thread(s) per core	1
Core(s) per socket	4
Socket(s)	2
NUMA node(s)	1
Total RAM Memory	4GB DDR4 @ 2400 MHz
Hypervisor vendor	VMware
Virtualization type	Full
OS	Ubuntu 16.04.6 LTS
Version	16.04.6 LTS (Xenial Xerus)

Table 5.2. Web Server Specs

For solution development, language model training and pre-training we took advantage of Google Colaboratory coding environment. Colaboratory, or *Colab* for short, is a free to use Google product that allows to write and execute code on a hosted Jupyter notebook in Python. More technically is an environment that provides free access to cloud TPUs and GPUs computational resources. More information about the service may be found at `https://research.google.com/colaboratory/faq.html`. Well suited especially to machine learning, education and data analysis, these free resources are of course not unlimited regarding hardware and time (up to a maximum continuous utilization of 12 hours per session) of utilization.

CPU Model	Intel(R) Xeon(R) CPU @ 2.00GHz
GPU	Nvidia K80
GPU Max Available Memory	13.3 GB
RAM Memory Available	25 GB (Max)

Table 5.3. Google Colaboratory Specs Feb 2020

5.3 Software

5.3.1 Development tools

The programming language used is Python version 3.6 [28], which is often chosen as an interpreted, object-oriented, interactive language for research. Incorporating very high level dynamic data types, classes and dynamic typing, it combines its multiple features with clear syntax and a lot of versatility.

To facilitate and better organize development we adopted an open-source web application named Jupyter Notebook [29] that offers the possibility to run different programming languages. One of its main features is its architecture that let us handle a complete installation of a kernel, libraries and tools.

As editor of source code we used Visual Studio Code, in addition to Google Colaboratory, a multi-platform development tool built by Microsoft that includes amongst its many features support for debugging, integrated Git management, IntelliSense and code refactoring. It is based on Electron, an open-source framework that combines Chromium rendering, an open-source web browser, and Node.js runtime.

5.3.2 Libraries and Frameworks

Web-based Annotation Tool

In order to create and manage the web-based annotation tool application we employed mainly Vue.js, Node.js and Docker by taking as a starting point doccano [16] open source project.

Vue.js, also known simply as Vue, is a JavaScript open-source framework for front-end web applications. It is based on Model-view-viewModel configuration and presents an architecture adoptable in a incremental way that focuses on declarative rendering and components composition. We used Vue as core front-end component.

Node.js, or simply Node, is a Input/Output event-oriented JavaScript open-source runtime for JavaScript code execution. In addition to being multi-platform, its event-oriented architecture focuses on asynchronous I/O with the aim of optimising web applications throughput and scalability. We used Node as main application management tool.

Docker is an open-source project designed to automate application deployment using software containers by offering OS-level virtualization. It is an instrument able to create a stand-alone working package of an application and its dependencies in a virtual container that can be executed in any machine running the Docker engine. We used Docker to envelop our web application in a deployable package ready to run on healthcare facility servers.

spaCy

To develop a light computational model solution we used *spaCy* [17] version 2.2 as a core element. Mostly written by Matthew Honnibal and Ines Montani, spaCy is an open-source library for advanced industrial-strength natural language processing whose main objective is to build solutions for production usage. Written in Python and

Cython [1] and managed by Explosion software company [2], it supports deep learning workflows with a good balance between performance and speed. It uses a custom machine learning open-source library named Thinc [3] to develop statistical model core components like Maxout units, affine layers, residual networks etc. This library, amongst its many functionalities, offers production-ready customizable implementations for:

- Part-of-speech (POS) Tagging
- Labelled Dependency Parsing
- Syntax-driven Sentence Segmentation
- Named Entity Recognition
- Non-destructive Tokenization
- Sentence or Text Classification
- Word similarity measuring (through word vectors or tensors)
- Lemmatization
- Sentence Boundary Detection
- Entity Linking
- Rule-based Matching
- Serialization
- Built in visualizers for syntax and NER

[1]Cython: https://en.wikipedia.org/wiki/Cython

[2]Explosion: https://explosion.ai/

[3]Thinc: A machine learning library optimized for CPU usage and deep learning with text input https://github.com/explosion/thinc

The library, dissimilar to a platform, does not aim to provide a software as a service neither an out-of-the-box tool in fact being open source it pushes strongly for customization to build NLP applications with its pipelines. spaCy amongst its downloadable packages provides pre-trained statistical language models for multiple supported languages:

- English
- German
- French
- Spanish
- Portuguese
- Italian
- Dutch
- Greek
- Norwegian Bokmål
- Lithuanian
- Russian

Unfortunately, beside English and few others, most of pre-trained models like the Italian one are available solely in the so called "small" version meaning that they do not carry with them word vectors but with only with pre-trained tensors to compute word similarity and contextual embeddings. This important aspect brought us to import Italian word vectors from other sources developed with word2vec, GloVe and fastText algorithms on the Italian Wikipidia, expand their vocabulary with the domain specific words residing in GIC post summaries and finally merge them with a language model using the *gensim* library.

Gensim

Gensim [30] that equals "Generate Similar" is an open-source library written in Python and Cython. Its aim is to provide to natural langage processing and information retrieval community tools for unsupervised topic modeling, similarity retrieval and document indexing for large text corpora. This library is designed around scalability and for this purpose it uses only memory-independent algorithms (able to process input larger than available RAM memory by streaming and out-of-core techniques) in order to handle large text collections. Amongst many others, Gensim includes parallelized implementations of word2vec, fastText and GloVe algorithms. We used this library to manipulate and expand Italian word vectors with healthcare domain specific vocabulary.

PyTorch

PyTorch [31] is one of the most used frameworks for machine learning applications such as computer vision and natural language processing. PyTorch is an open source library based on Torch library [4], developed primarily by Facebook's AI Research lab (FAIR). Beside being Python-first, it provides two high-level features:

- Tensor computation with strong GPU acceleration
- Deep neural networks built on a tape-based autograd system

Scientific computations such as slicing, math operations, reduction, linear algebra are accelerated through optimized tensor routines that can either run on CPUs or GPUs. PyTorch's tape-based autograd techniques let the user change network configurations without lag or overhead granting great speed and flexibility by using dynamic computational graph paradigm to represent and build neural networks.

[4]Torch machine learning library: https://github.com/torch/torch7

Every forward call generates a computational graph on the fly that can be easily edited and managed if a configuration change is needed. This library is the core engine of many deep learning software like HuggingFace's Transformers.

Huggingface's Transformers

In order to manage the solution involving a transformer model from the BERT family we leaned on PyTorch framework and the well known *huggingface-trasformers* library [32], also known as pytorch-transformers, which offers an easy access to state-of-the-art NLP models. Progress in today's NLP research has seen many transfer learning techniques with large-scale language models based on Transformer architecture. These models often share the same pattern of training on general purpose tasks and fine-tuning on domain specific problems. Generate these general-purpose models is costly but, for many, to even implement pre-trained ones is often difficult and time-consuming due to multiplicity of frameworks and code bases. Luckily many libraries like huggingface-transformers aim to solve this issue. This library includes many general-purpose architectures (Complete documentation available at `https://huggingface.co/transformers/`) with deep interoperability between TensorFlow 2.0 [5] and PyTorch frameworks:

- BERT (from Google)
- GPT (from OpenAI)
- GPT-2 (from OpenAI)
- Transformer-XL (from Google/CMU)
- XLNet (from Google/CMU)
- XLM (from Facebook)

[5]TensorFlow framework: https://www.tensorflow.org/

- RoBERTa (from Facebook)
- DistilBERT (from HuggingFace)
- CTRL (from Salesforce)
- CamemBERT (from Inria/Facebook/Sorbonne)
- ALBERT (from Google Research/Chicago TTI)
- T5 (from Google AI)
- XLM-RoBERTa (from Facebook AI)
- MMBT (from Facebook)
- FlauBERT (from CNRS)
- And other community models shared by the community.

Many models initially written in Tensorflow and other frameworks are available under the same library.

To speed up the implementation process of the code even more we exploited a new available library named *SimpleTransformers* [6] whose goal is to act as an interface to pytorch-transformers by providing a quick setup and management of the most used huggingface-transformers features.

[6]SimpleTransformers: https://github.com/ThilinaRajapakse/simpletransformers

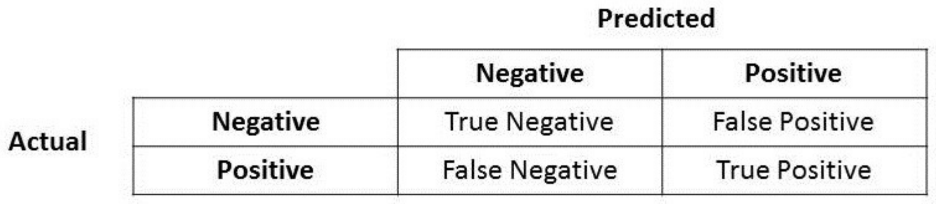
Chapter 6

Results

In this chapter we explain the metrics used to evaluate the models followed by the obtained results for the task.

6.1 Metrics

A core task in building machine learning models is evaluation. While building predictive models we use feedback from metrics to make improvements. The problem that we face is a multi-label classification problem therefore for each label we compute a confusion matrix of the obtained predictions.



		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

Figure 6.1. Confusion Matrix

To obtain useful metrics out of the confusion matrix we compute averages. There are two main types of averages:

- Macro Averages
- Micro Averages

Macro-averages method computes the overall average of each class label measured metric while Micro-averages sums up the individual True Positives, False Positives, False Negatives and only then computes the metrics. The kinds of measurements that we use follow the Micro-Averages method. From each label class confusion matrix we compute metrics per label and, through micro-average method, the overall results for all classes. We implement three kind of metrics that are:

- Precision
- Recall
- F-Score

Precision is a measure of exactness or quality, is used to determine how many actual positives a model correctly predicted out of the predicted positives.

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

Recall is a measure of completeness or quality, is used to determine *how complete the results are.*

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

F-score, or F1 Score, is the harmonic mean of precision and recall that combines them together.

$$F1 = 2\frac{Precision \times Recall}{Precision + Recall}$$

6.1.1 K-Fold Cross Validation

While building supervised statistical models, we use cross validation in order to avoid overfitting by learning model's parameters and computing results on the very same data used for training. Cross validation is a statistical method used to test models on unseen data (test set) by holding out a part of the available dataset during training. It is a procedure often used on limited data sample, it has a single parameter called k that indicates the quantity of subgroups or folds into which the sample of input data is to be divided. Then for each of the k *folds* the model is trained using $k-1$ of the folds as training data and the remaining one for validation. Finally the resulting performance is the average of the values computed for each iteration in the loop.

$$Performance = \frac{1}{k}\sum_{i=1}^{k} Performance_i$$

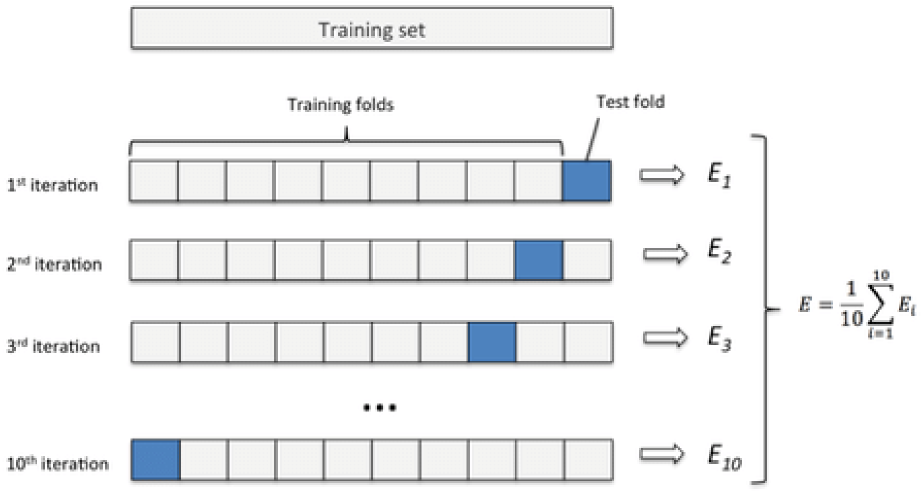


Figure 6.2. 10-Fold Cross Validation [33]

6.2 Hybrid Model Results

The light computational model based on the fast convolutional neural network coupled with domain rule-based deductive knowledge gave us interesting results given the limited training data. One of the best aspect of this approach is that it provides a good balance between speed and performance, in fact the training time is limited and the performances are comparable to the heavy computational results. To compute the results that we present, we selected the best parameters we found using a k-fold cross validation strategy. In details we selected:

- **K** of k-fold cross validation: 10
- **Epochs**: 25
- **Optimizer**: Adam Optimization Algorithm [34]
- **Learning Rate (Alpha)**: 0.001
- **Beta 1**: 0.9
- **Beta 2**: 0.999
- **Epsilon**: 1e-08
- **Ridge regression (L2) Regularization Penalty**: 1e-06
- **Max Norm Constraint**: 1.0
- **Width of Embeddings layers**: 128
- **Rows in Embedding Tables**: 7500

In order to pre-train the token-to-vector layer and doing so create weights usable for model training, we tried various Italian word vectors (computed via word2vec and GloVe algorithms) but the one that gave us the best results was computed with fastText algorithm. After enlarging Italian word vectors with domain specific clinical words, we trained the Convolutional Neural Network (CNN) part by chaining an output layer on top of the token-to-vector input model and training it as a masked language model with a L2 loss function in order to predict vectors which match the pre-trained ones. The pre-train phase on unlabelled text was carried out with following parameters:

- **Epochs**: 1000
- **Width of CNN layers**: 96
- **Depth of CNN layers**: 4
- **Maxout unit pieces**: 3
- **Window size of CNN layers**: 1
- **Embedding rows**: 2000
- **Dropout rate**: 0.2
- **Random seed**: 0

Once obtained weights to initialize the CNN through this pre-training procedure, we trained the model classifier for named entity recognition using labeled training data. For evaluation we used a ten fold cross validation approach splitting data in training and test set by document.

Chronologically, the first model we used was based on a pure inductive approach that implements the previously described CNN and a multi-layer perceptron for classification as key components. However, due to the narrow amount of data examples available, the results were limited. To overcome this issue we used a deductive approach by writing a set of handwritten rules based on specific knowledge related to the domain. The fusion of the inductive statistical model with rule-based deductive knowledge led to the birth of a hybrid model.

The results obtained for this multi-token classification problem are shown in the tables below:

	Inductive	**Deductive**	**Hybrid**
Precision	0.468254	0.39613	0.443389
Recall	0.242466	0.635711	0.767123
F-Score	0.319495	0.480862	0.561967

Table 6.1. Inductive, Deductive and Hybrid Models Overall Results

	Precision	**Recall**	**F-Score**
Biopsia	0.204678	0.406977	0.272374
BRCA	0	0	0
ER	0.691176	0.824561	0.752
Grado	0.376344	0.897436	0.530303
HER 2	0.385366	0.877778	0.535593
Invasione vascolare	0.714286	0.897436	0.795455
Ki67	0.571429	0.615385	0.592593
M	0.25	1	0.4
Macrometastasi	0.196429	1	0.328358
Margine	0.8	0.780488	0.790123
Micrometastasi	0.5	1	0.666667
N	0.553191	0.8125	0.658228
Neoadiuvante	0.138889	0.714286	0.232558
PGR	0.698413	0.785714	0.739496
RM Mam	0.465517	0.771429	0.580645
Stadio	0.686275	0.921053	0.786517
T	0.442478	0.769231	0.561798
Visita oncogeriatrica	0.4	0.571429	0.470588

Table 6.2. Hybrid Model Results by Label

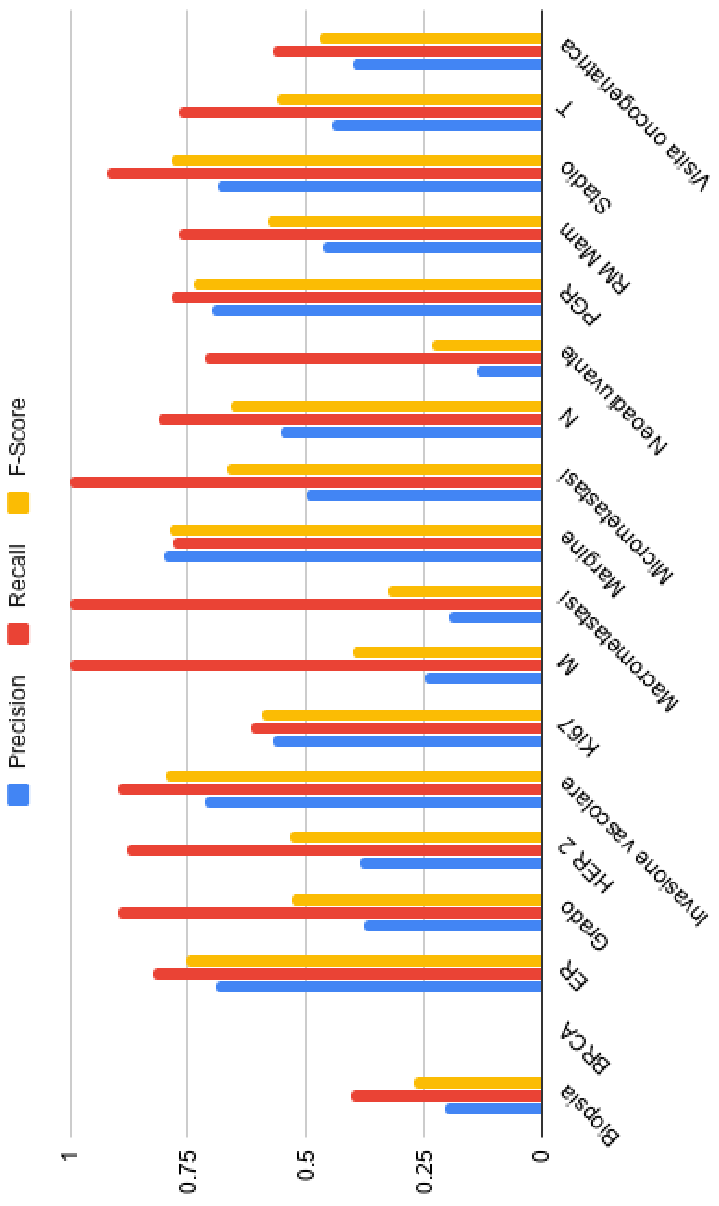


Figure 6.3. Hybrid Model Results by Label

As we can see from the results tables, this model based on a fast and light convolutional neural network obtained some interesting outcomes. Despite the limited data set in our possession, the model has achieved some average acceptable scores on the recall measure which has a critical importance in the healthcare domain where each false negative could lead to costly errors. Contrariwise the precision measure obtained low scores which may due to the unbalanced input data over the chosen class labels and almost certainly to its limited number of available examples.

6.3 BERT Model Results

The heavier computational model from the BERT family gave us better results on our limited training data. Due to their high computational requirements, the training was performed on GPUs provided by the Google Colaboratory environment. Even with our limited dataset, the training times and costs were high which allow us to easily deduct that these models are not suitable for an evolutionary solution where the model is trained multiple times with new data generated by the web-based annotation tool. Nevertheless in order to obtain a baseline, we tried to adopt multiple pre-trained models from the BERT family, from official multi-lingual releases and Italian research community. As often unfortunately happens, while many official models often offer a wide range of releases for the English language few are delivered with support of Italian or multi-lingual. The models that we explored for this named entity recognition activity are the following:

- **BERT base multilingual cased**
- BERT base multilingual uncased
- DistilBERT base multilingual cased
- UmBERTo
- GilBERTo

Despite UmBERTo and GilBERTo being pre-trained models provided by the Italian research community focusing on the Italian language, the one who gave us the best result was the BERT base multilingual cased model. This model supports 104 languages, amongst which Italian, and it is composed of:

- 12 transformer layers
- hidden size of 768
- 12 self-attention heads
- 110 millions of parameters

The Best configuration that we found during our experiments is the following:

Epochs	3
Weight decay	0
Maximum sequence length	192
Train batch size	8
Evaluation batch size	8
Learning rate	4e-5
Adam Epsilon	1e-8
Maximum gradient norm	1.0
Do lower case	Disabled
Half precision floating point	Enabled
Half precision fp optimization level	O1

Table 6.3. BERT Model Configuration

The results obtained for this multi-token classification problem are shown in the tables below:

Precision	0.649017
Recall	0.587671
F-Score	0.616822

Table 6.4. BERT Model Overall Results

	Precision	**Recall**	**F-Score**
Biopsia	0.522936	0.662791	0.584615
BRCA	0	0	0
ER	0.884615	0.807018	0.844037
Grado	0.611111	0.564103	0.586667
HER 2	0.816327	0.444444	0.57554
Invasione vascolare	0	0	0
Ki67	0.764706	0.75	0.757282
M	0	0	0
Macrometastasi	0.727273	0.727273	0.727273
Margine	0.35	0.170732	0.229508
Micrometastasi	0	0	0
N	0.620253	0.765625	0.685315
Neoadiuvante	0	0	0
PGR	0.849057	0.803571	0.825688
RM Mam	0.6	0.685714	0.64
Stadio	0.823529	0.736842	0.777778
T	0.482759	0.646154	0.552632
Visita oncogeriatrica	0	0	0

Table 6.5. BERT Model Results by Label

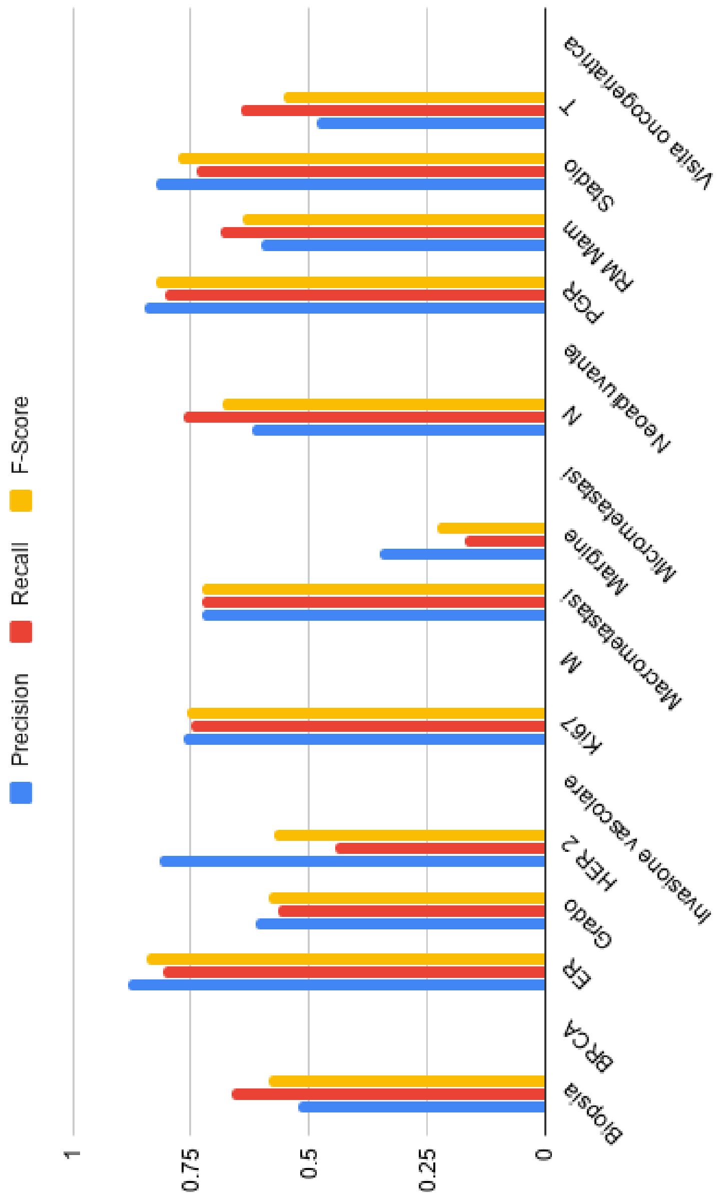


Figure 6.4. BERT Model Results by Label

As we can see from the result boards by implementing the BERT model, precision and recall are way closer to each other however it fails to recognise class labels without an enough number of examples. As expected, the resulting scores obtained with the BERT model are higher than the ones obtained with the convolutional neural network. One thing that is very interesting is that the difference between the heavy computational expensive BERT model and the hybrid model is slighter than expected which is very remarkable. This is probably due the fact that, to overcome the lack of available examples, we enriched the CNN-based statistical model with deductive rule-based knowledge.

Chapter 7

Conclusions and Future Works

This master thesis work acted as an initial thrust to my studies in the astonishing and intriguing world of natural language understanding. Taking this work as an example it is evident that we have scratched only the surface of what we can do with the big amount of digital data nowadays available but not yet used. Natural Language Processing has very distant roots in human history but in recent years it discovered a new wave of innovation. Since deep learning have been firstly applied to unstructured text problems, new techniques and discoveries are revolutionizing this field of research. Year by year, paper by paper new models raise the bar of natural language understanding higher and higher. There is still a significant gap between what is now available from the latest research and what is actually implemented in real-world challenges.

In this master thesis work we presented our solution for the semantic annotation of clinical notes and obtained valuable results. We introduced the world of natural language processing and its sub-task of information extraction called Named Entity Recognition. After that we illustrated the different approaches and several techniques applied to solve this kind of tasks in the recent years.

By using this knowledge as a firm baseline, we initiated our journey between the various possible approaches available in literature for named entity recognition in order to select and forge a model suited around a specific problem regarding the healthcare world. Through collaboration with the healthcare environment and domain experts we obtained a manually annotated data set of clinical notes regarding post surgical breast cancer cases. The manual annotation phase alongside the redacting of a class taxonomy document may appear an easy and consolidated process but in the aftermath we can easily assert that it has been one of the most expensive and time consuming, this is due to the difficulties in combining different visions between many actors carrying distinct backgrounds not related to the world of information technology to reach the very same and unique goal, like domain experts which belong to the surgical world or data managers specialized in administrative tasks. As a consequence, consolidating a class taxonomy document and guide domain experts toward a correct way of annotating took us many iteration and effort but it is an initial cost hardly avoidable that we can represent as a cold start problem. This non-trivial manual annotation phase experience enriched every one of us and will surely come in handy in future works.

The collected data set is comprised of a collection of reports from the interdisciplinary group that is assembled together in order to effectively cure the patient. Having to face a wide range of original label classes that do not overlap at all with those normally studied in the literature, we proposed an hybrid solution that aims to a good balance between performance and costs by merging together a deductive rule-based approach with a convolutional neural network based machine learning model computationally light enough to fit our technical constraints.

However, In order to obtain a baseline for results comparison with deployable state-of-the-art models, we explored the path leading to a transformer-based solution involving the models from the BERT family exploiting transfer learning applied to named entity recognition. As expected the results obtained from the hybrid convolutional neural network approach are inferior to the ones from the BERT model but the interesting point is that the difference is thin nevertheless the latter one is characterized by way higher training costs and scalability problems.

This experiment showed that named entity recognition applied onto domain specific text data with original classes is a hard problem for which machine learning models requires an adequate quantity of labeled training data set which is unfortunately often not available nor always possible to obtain. To overcome these lacks while maintaining a computationally feasible model capable of running in general purpose machines a hybrid approach that includes deductive rule-based knowledge is needed.

7.1 Future Works

For what concern the Named Entity Recognition, the goal is to find new ways to train bi-directional transformers models and embeddings able to completely represent the meaning of the word given its context without making computational costs too heavy for real case applications. Furthermore obtain more resources and tools ready to use in real world scenarios for other languages beside English, such as Italian. Recent researches in this field are finding new paths to train heavy computational models like BERT in a more efficient a precise way like RoBERTa [35] or Cloze-driven [36] approaches.

The solution presented in this work has surely wide margins of improvements, one of the main concerns regarding this task is indeed the size of the data set used to train the models which, due to strict deadlines of the project, is undoubtedly small and insufficient for many label classes. In the near future the aim is to include multiple breast units in the project in order to obtain a way bigger data set and a larger workforce of multiple manual domain expert annotators. The goal is certainly to have a solid and large base of annotated data usable as gold standard with which train a generalized machine learning model, with a good trade-off between performance and costs, deployable directly in healthcare facilities and capable of automatizing the semantic annotation of clinical notes.

Bibliography

[1] Wikipedia. *Natural Language Processing — Wikipedia, The Free Encyclopedia*. [Online; accessed 08-Mar-2020]. 2020. URL: https://en.wikipedia.org/wiki/Natural_language_processing.

[2] Wikipedia. *Named Entity Recognition — Wikipedia, The Free Encyclopedia*. [Online; accessed 08-Mar-2020]. 2020. URL: https://en.wikipedia.org/wiki/Named-entity_recognition.

[3] *WordLift vocabulary, named entity recognition*. Accessed on 08/03/2020. URL: https://wordlift.io/blog/en/entity/named-entity-recognition/.

[4] Tomas Mikolov et al. *Distributed Representations of Words and Phrases and their Compositionality*. 2013. arXiv: 1310.4546 [cs.CL].

[5] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL].

[6] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. «GloVe: Global Vectors for Word Representation». In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[7] Piotr Bojanowski et al. *Enriching Word Vectors with Subword Information*. 2016. arXiv: 1607.04606 [cs.CL].

[8] Armand Joulin et al. *Bag of Tricks for Efficient Text Classification*. 2016. arXiv: 1607.01759 [cs.CL].

[9] Basant Agarwal et al. *Deep Learning-Based Approaches for Sentiment Analysis*. Springer, 2020.

[10] Wikipedia. *Softmax function — Wikipedia, The Free Encyclopedia*. [Online; accessed 14-Mar-2020]. 2020. URL: https://en.wikipedia.org/wiki/Softmax_function.

[11] RInterested. *Softmax function*. [Online; accessed 14-Mar-2020]. 2020. URL: http://rinterested.github.io/statistics/softmax.html.

[12] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2014. arXiv: 1409.0473 [cs.CL].

[13] Ashish Vaswani et al. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL].

[14] Yoon Kim. «Convolutional Neural Networks for Sentence Classification». In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751. DOI: 10.3115/v1/D14-1181. URL: https://www.aclweb.org/anthology/D14-1181.

[15] *American Joint Commission on Cancer; 2017*. Springer International Publishing, 2017.

[16] Hiroki Nakayama et al. *doccano: Text Annotation Tool for Human*. Software available from https://github.com/doccano/doccano. 2018. URL: https://github.com/doccano/doccano.

[17] Matthew Honnibal and Ines Montani. «spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing». https://spacy.io/. 2017.

[18] Siti Mariyah et al. «Reveal the customer behavior from business sms text using named entity recognition». In: (July 2019). DOI: 10.24507/icicel.13.08.653.

[19] Wikipedia. *Lexical analysis — Wikipedia, The Free Encyclopedia*. [Online; accessed 19-Feb-2020]. 2020. URL: https://en.wikipedia.org/wiki/Lexical_analysis.

[20] Dan Svenstrup, Jonas Meinertz Hansen, and Ole Winther. *Hash Embeddings for Efficient Word Representations*. 2017. arXiv: 1709.03933 [cs.CL].

[21] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG].

[22] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. arXiv: 1607.06450 [stat.ML].

[23] *Layer Normalization*. 2018. URL: https://mlexplained.com/2018/01/13/weight-normalization-and-layer-normalization-explained-normalization-in-deep-learning-part-2/.

[24] Ronan Collobert et al. *Natural Language Processing (almost) from Scratch*. 2011. arXiv: 1103.0398 [cs.LG].

[25] Sachin Kumar and Yulia Tsvetkov. *Von Mises-Fisher Loss for Training Sequence to Sequence Models with Continuous Outputs*. 2018. arXiv: 1812.04616 [cs.CL].

[26] Jacob Devlin et al. «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding». In: *arXiv preprint arXiv:1810.04805* (2018).

[27] Yonghui Wu et al. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. 2016. arXiv: 1609.08144 [cs.CL].

[28] Python Core Team (2015). *Python: A dynamic, open source programming language.* Python Software Foundation. 2015. URL: https://www.python.org/..

[29] Thomas Kluyver et al. «Jupyter Notebooks – a publishing format for reproducible computational workflows». In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. Ed. by F. Loizides and B. Schmidt. IOS Press. 2016, pp. 87 –90.

[30] Radim Řehůřek and Petr Sojka. «Software Framework for Topic Modelling with Large Corpora». English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. http://is.muni.cz/publication/884893/en. Valletta, Malta: ELRA, May 2010, pp. 45–50.

[31] Adam Paszke et al. «PyTorch: An Imperative Style, High-Performance Deep Learning Library». In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. https://pytorch.org/. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[32] Thomas Wolf et al. «HuggingFace's Transformers: State-of-the-art Natural Language Processing». In: *ArXiv* abs/1910.03771 (2019).

[33] Juan Buhagiar et al. «Automatic Segmentation of Indoor and Outdoor Scenes from Visual Lifelogging». PhD thesis. June 2017.

[34] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. arXiv: 1412.6980 [cs.LG].

[35] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL].

[36] Alexei Baevski et al. *Cloze-driven Pretraining of Self-attention Networks*. 2019. arXiv: 1903.07785 [cs.CL].