UNIVERSITÀ
DEGLI STUDI
FIRENZE

PhD Program in Smart Computing
Dipartimento di Ingegneria dell'Informazione (DINFO)

Classification of cancer pathology reports with Deep Learning methods

Stefano Martina

Dissertation presented in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Smart Computing

PhD Program in Smart Computing
University of Florence, University of Pisa, University of Siena

Classification of cancer pathology reports with Deep Learning metho

Stefano Martina

Advisor:

Prof. Paolo Frasconi

Head of the PhD Program:

Prof. Paolo Frasconi

Evaluation Committee:
Prof. Vassilis Katsouros, *Institute for Language and Speech Processing (ILSP)*
Prof. Foteini Simistira Liwicki, *Luleå University of Technology*

XXXII ciclo — October 2019

I can live with doubt, and uncertainty, and not knowing. I think it's much more interesting to live not knowing than to have answers which might be wrong. I have approximate answers, and possible beliefs, and different degrees of certainty about different things, but I'm not absolutely sure of anything. There are many things I don't know anything about, such as whether it means anything to ask "Why are we here?" I might think about it a little bit, and if I can't figure it out then I go on to something else. But I don't have to know an answer. I don't feel frightened by not knowing things, by being lost in the mysterious universe without having any purpose — which is the way it really is, as far as I can tell. Possibly. It doesn't frighten me.

Richard P. Feynman

Acknowledgments

First of all, I would like to express my gratitude to my advisor Prof. Paolo Frasconi for his expert guidance, for the important insights he shared about the field of AI and the always new and inspiring inputs about research.

Second, I would like to thank Leonardo Ventura, Prof. Gianni Amunni, and ISPRO for giving me the opportunity to work with their data and making this work of thesis possibile.

Thanks to Prof. Søren Brunak and Prof. Beatrice Lazzerini for helping me advancing my research.

I'm grateful to Prof. Enrico Vicario for introducing me to this doctorate and to Prof. Simone Marinai for his precious advice.

A special thanks goes to the people who shared the AI-Team lab and the Ph.D life with me. In alphabetical order: Daniele Baracchi, Samuele Capobianco, Alessandro Lazzeri, Francesco Orsini, Giulia Pellegrini, Dasara Shullani, Alessandro Tibo, La Ode Toresano, Amin Zadenoori, and Zahra Ziran.

I would like to thank all my friends that were always there to support me, root for me and cheer me up in moments of crisis and doubt. Special mentions: Andrea Benassai, Vito Bonelli, Giulia Bondielli, Giulia Bono, Tiberio Uricchio. Thanks to my dear family Mauro, Rosa, and Simone for their presence and affection throughout my whole life. Thanks to Fede for all the help that she gave me in this thesis and in my whole life and for loving me dearly. Last but not least, thanks to Spallina for accompanying me from the start to the end of my academic path.

Abstract

Natural Language Processing (NLP) is a discipline that involves the design of methods that process text. Deep learning, and Machine Learning (ML) in general, is the discipline that studies and implements methods that learn to make predictions from data. In the last years, many different ML methods have been presented in the context of NLP. In this work we focused in particular on text classification methods. Cancer registries collect pathology reports from clinical data sources and combine them with administrative data sources to identify cancer diagnoses in a specific area. Here we present a large scale study on deep learning methods applied to cancer pathology reports in Italian language. In this study we developed several classifiers to predict topography and morphology ICD-O codes. We compared classic machine learning approaches, i.e. Support Vector Machine (SVM), with recent deep learning techniques, i.e. Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). Furthermore, we compared recent attention-based and hierarchical techniques, e.g. Bidirectional Encoder Representations from Transformers (BERT), with a more simple hard attention method, showing that the latter is enough to perform slightly better in this specific domain.

Contents

Chapter 1

Introduction

1.1 Cancer registries

Cancer is a major concern worldwide, as it decreases the quality of life and leads to premature mortality. In addition it is one of the most complex and difficult-to-treat diseases, with significant social implications, both in terms of mortality rate and in terms of costs associated with treatment and disability [1, 15, 42, 44]. Measuring the burden of disease is one of the main concerns of public healthcare operators. Suitable measures are necessary to describe the general state of population's health, to establish public health goals and to compare the national health status and performance of health systems across countries. Furthermore, such studies are needed to assess the allocation of health care and health research resources across disease categories and to evaluate the potential costs and benefits of public health interventions [5].

Cancer registries emerged during the last few decades as a strategic tool to quantify the impact of the disease and to provide analytic data to healthcare operators and decision makers. Cancer registries use administrative and clinical data sources in order to identify all the new cancer diagnoses in a specific area and time period and collect incidence records that provide details on the diagnosis and the outcome of treatments. Mining cancer registry datasets can help towards the development of global surveillance programs [47] and can provide important insights such as survivability [14]. Although data analysis software would best operate on structured representations of the reports, pathologists normally enter data items as free text in the local country language. This requires intelligent algorithms for medical document information extraction, retrieval, and classification, an area that has received significant attention in the last few years (see, e.g., [38] for a recent account and [53] for the specific case of cancer).

1.2 ICD-O

Pathology reports can be classified according to codes defined in the International Classification of Diseases for Oncology, third edition (ICD-O-3) system [20], a specialization of the ICD for the cancer domain which is internationally adopted as the standard classification for topography and morphology [19]. The development of text analysis tools specifically devoted to the automatic classification of incidence records according to ICD-O3 codes has been addressed in a number of previous papers. However, these works have either focused on reasonably large datasets but using simple linear classifiers based on bag-of-words representations of text [26, 27], or applied recent state-of-the-art deep learning techniques [21, 41] but using smaller datasets and restricted to a partial set of tumors. Additionally, the use of deep learning techniques usually requires accurate domain-specific word vectors (embeddings of words in a vector space) that can be derived from word co-occurrences in large corpora of unlabeled text [16, 36, 39]. Large medical corpora are easily available for English (e.g. PubMed) but not for other languages.

A topographical ICD-O-3 code is structured as *Cmm.s* where *mm* represent the main site and *s* the subsite. For example, *C50.2* is the code for the upper-inner quadrant (*2*) of breast (*50*).

A morphological ICD-O-3 code is structured as *tttt/b* where *tttt* represent the cell type and *b* the tumor behavior (benign, uncertain, in-situ, malignant primary site, malignant metastatic site). For example, *8140/3* is the code for an adenocarcinoma (*adeno 8140*; *carcinoma 3*).

1.3 Existing works on ICD-O

Early works for ICD-O3 code assignment were structured on rule-based systems, where the code was assigned by creating a set of handcrafted text search queries and combining results by standard Boolean operators [13]. In order to prevent spurious matches, rules need to be very specific, making it very difficult to achieve a sufficiently high recall on future (unseen) cases.

A number of studies reporting on the application of machine learning to this problem have been published during the last decade. Direct comparisons among these works are impossible due to the (not surprising) lack of standard publicly available datasets and the presence of heterogeneous details in the settings. Still, we highlight the main differences among them in order to provide some background. In [26], the authors employed support vector machine (SVM) and Naive Bayes classifiers on a small dataset of 5 121 French pathology reports and a reduced number of target classes (26 topographic classes and 18 morphological classes), reporting an accuracy of 72.6% on topography and 86.4% on morphology with SVM. A much larger

dataset of 56 426 English reports from the Kentucky Cancer Registry was later employed in [27], where linear classifiers (SVM, Naive Bayes, and logistic regression) were also compared but only on the topography task and using 14, 42, and 57 classes. The authors reported a micro-averaged F1 measure of 90% when using SVM with both unigrams and bigrams. Still, the bag-of-words representations used by these linear classifiers do not consider word order and are unable to capture similarities and relations among words (which are all represented by orthogonal vectors). Deep learning techniques are known to overcome these limitations but were not applied to this problem until very recently. In [41], a Convolutional Neural Network (CNN) architecture fed by word vectors pretrained on PubMed was applied to a small corpus of 942 breast and lung cancer reports in English with 12 topography classes; the authors demonstrate the superiority of this approach compared to linear classifiers with significant increases in both micro and macro F1 measures. In [21], the same research group also experimented on the same dataset using Recurrent Neural Networks (RNNs) with hierarchical attention [52], obtaining further improvements over the CNN architecture.

1.4 Motivation and contributions

With this work we want to answer to some questions.

Q1 At the best of our knowledge, there are no large scale studies on deep learning method applied to pathology reports. Cancer registries collect large amount of records and invest time labeling them with topographical and morphological codes. Existing works focus on ICD-O classification of small datasets or using few classes. We want to make a step further applying deep learning techniques to a large scale dataset.

Q2 We want to apply novel deep learning techniques, like attention models and BERT, to cancer pathology reports text classification. We want to propose experiments to assess how they perform on this kind of data.

Q3 It is not yet clear from the literature if, in this domain, bag-of-words techniques are inferior to deep learning techniques. The structure of the sentences in italian-language cancer pathology reports differs substantially from the structure of the text where RNN methods are usually employed. The records often do not present the usual subject-verb-object structure.

Q4 It is not clear if the use of novel attention-based and hierarchical techniques represents an improvement with respect to simpler RNN methods on this kind of text.

Q5 It is not clear the contribution and the applicability of unsupervised learning techniques to uncommon text corpora. Usually, word vectors are trained on long english text corpora, taken from Wikipedia, Gigaword, or similar. Our documents not only are in a different language, but they also have a different structure.

Q6 We want to evaluate the possibility to give some interpretation to the notoriously uninterpretable deep learning models.

Chapter 2

Machine Learning

2.1 Machine Learning Theory

ML is a branch of Artificial Intelligence (AI) where the focus is to develop systems that learn from data. ML is a field that is located in the intersection of different disciplines:

statistics deals with the uncertainty of the world and, as we will see, ML can be framed as a probability estimation;

data science is the science that interprets and studies the information contained in the data and how it can be used;

optimization theory is the mathematical branch that defines methods to optimize functions. ML can be seen as an optimization problem;

computer science is the science that studies algorithms and their complexity;

computer engineering focuses on the development of efficient software.

In order to learn we must provide some data. The data can be of different kinds. Based on the type of data and on the problem that we need to resolve, we can denote three different types of ML: supervised, unsupervised, and reinforcement learning.

Supervised Learning

In supervised learning we have a dataset S of samples $\boldsymbol{x}_i$ each one labeled with $\boldsymbol{y}_i$:

$$S = ((\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)),$$

where $\boldsymbol{x}_i \in \mathbb{X}$ and $\boldsymbol{y}_i \in \mathbb{Y}$. Usually, $\mathbb{X}$ is a space of integer or real tensors of certain dimensionality, and $\mathbb{Y}$ is a space of integer vectors for classification tasks and

real vectors for regression tasks. In binary classification tasks $\mathbb{Y} = 0, 1$, in multilabel classification $\mathbb{Y} = 0, 1^k$ with k possible labels, and in multiclass classification $\mathbb{Y} = 0, 1^k$ with only one value equal to 1 and the rest equal to 0. The samples $\boldsymbol{x}_i$ of a dataset must be drawn from the same unknown distribution $\boldsymbol{x}_i \sim \mathcal{D}$. The relationship between the samples and the labels is defined by an unknown labeling function $f : \mathbb{X} \to \mathbb{Y}$ such that for each sample $\boldsymbol{y}_i = f(\boldsymbol{x}_i)$.

A learning algorithm receives a training set S as input, and should output a predictor $h_S : \mathbb{X} \to \mathbb{Y}$ that minimizes the *prediction error*. For a generic predictor h, the prediction error is:

$$L_{\mathcal{D},f}(h) \stackrel{\text{def}}{=} \mathbb{P}_{\boldsymbol{x}\sim\mathcal{D}}[h(\boldsymbol{x}) \neq f(\boldsymbol{x})] \stackrel{\text{def}}{=} \mathcal{D}(\{\boldsymbol{x} : h(\boldsymbol{x}) \neq f(\boldsymbol{x})\}), \tag{2.1}$$

where, for $A \subset \mathbb{X}$, the probability $\mathcal{D}$ assign a likelihood $\mathcal{D}(A)$ of observing a value $\boldsymbol{x} \in A$. Given that both $\mathcal{D}$ and f are unknown, to find h_S the learning algorithm minimizes the *empirical prediction error* (or empirical risk):

$$L_S(h) \stackrel{\text{def}}{=} \frac{|i \in \{1, \ldots, n\} : h(\boldsymbol{x}_i) \neq \boldsymbol{y}_i|}{n}. \tag{2.2}$$

This learning paradigm of finding h_S is called Empirical Risk Minimization (ERM).

ERM rule may lead to *overfitting* if not restricted. A predictor can perform well over the training set S — having a low empirical error — but it can perform badly over the entire distribution $\mathcal{D}$ with a high prediction error. A solution is to apply ERM over a restricted search space. With $\mathbb{H}$ we denote the *hypothesis class* of the possible predictors $h \in HSet : \mathbb{X} \to \mathbb{Y}$. For a given class $\mathbb{H}$ and a training sample S, the $ERM_{\mathbb{H}}$ learner uses the ERM rule to choose a predictor $h_S \in \mathbb{H}$ with the lowest possible empirical error over S:

$$h_S = ERM_{\mathbb{H}}(S) \in \arg\min_{h\in\mathbb{H}} L_S(h).$$

We induce an *inductive bias* introducing restrictions over $\mathbb{H}$. Intuitively, choosing a more restricted hypothesis class better protect the model against overfitting but at the same time may cause a stronger bias.

2.2 Support Vector Machine (SVM)

SVMs [12] are supervised learning models commonly used in classification problems. The basic idea of linear SVMs is to find a hyperplane in the features space that separates examples belonging to different classes. At prediction time, the model classifies new examples based on their position relative to the hyperplane. For a given set of data, infinite separating hyperplanes could exist. SVMs search for the

maximum-margin separating hyperplane, i.e. the one with maximum distance from the nearest examples.

The basic SVM algorithm works only if the features belonging to the two classes are linearly separable. The *soft margin* method manages the cases where the classes are not linearly separable. A set of *slack* variables allows for a certain degree of misclassification. They allow for samples to be on the wrong side of the split.

The soft margin allows to separate data that is linearly separable except for some examples. Sets that are hardly not linearly separable sets can be approached using the *kernel trick*. If we call S the space of the features, the kernel trick works by finding a transformation $\phi : S \to V$ from the original space to a new space V where the examples are more easily separable. SVMs can then be used on the new space V.

2.3 Neural Networks

In our work, we used mainly a specific kind of RNNs: LSTM. RNNs are a class of ANNs where the connections are not only sequential from one layer to the following one, but they form loops instead.

Artificial Neural Network (ANN)

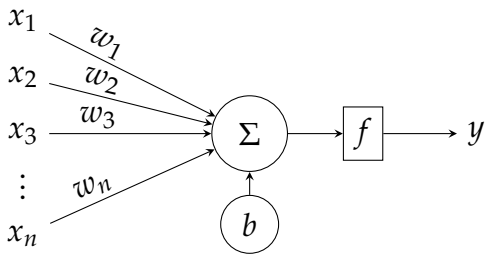


Figure 2.1: Artificial neuron.

An ANN is a model that performs elaborations in a way that mimics the brain functioning. The base unit is the Artificial Neuron (AN), also called perceptron, in fig. 2.1. It performs the weighted sum of the inputs $x_1, \ldots, x_n$, shifted by a bias b, followed by an activation function f. If we add a dummy input $x_0 = 1$ and rename the bias $b = w_0$, we can express the computation of the AN as in eq. (2.3):

$$y = f(\sum_{i=0}^{n} w_i x_i). \tag{2.3}$$

The activation function f can be of different types, the most common are:

- Softmax;

- ReLU;
- TanH;
- Sigmoid;
- Linear.

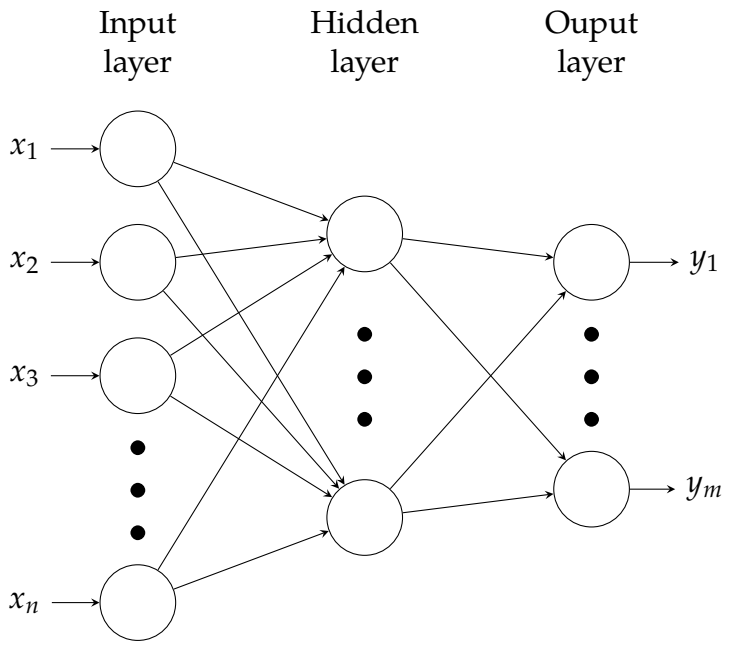


Figure 2.2: MLP with one hidden layer.

ANs are organized in network structures. The basic layout of an ANN is the Multilayer Perceptron (MLP) structured in layers like in fig. 2.2. Each AN of each layer is connected to all the outputs of the previous layer. The first layer is connected to the inputs of the ANN and the output of the last layer is also the output of the network. The execution of the MLP is feed forward:

1. the input values $x_{0,1}, \ldots x_{0,n}$ are presented to the network and they are the input of the first layer;
2. the computation is carried one layer at a time, where each neuron i of the layer l calculates the value $y_{l,i}$ of the intermediate output $y_{l,1}, \ldots, y_{l,m}$;
3. the intermediate output $y_{l,1}, \ldots, y_{l,m}$ of the layer l becomes the input $x_{l+1,1}, \ldots, x_{l+1,m}$ of the subsequent layer $l+1$, unless l is the last layer - in that case the output of l is the output $y_1, \ldots, y_m$ of the MLP.

The weights of the ANs are initialized to random values and, in order to have meaningful outputs, the ANN needs to be trained. In a supervised learning framework, the dataset is composed of the matrices $\boldsymbol{X}$ ($N \times n$) of the N inputs $x_{i,j}$ and $\boldsymbol{Y}$ ($N \times m$) of the corresponding outputs $y_{i,j}$. The training process is called *backpropagation* and it is organized in a sequence of phases. For each phase p there are two steps:

execution where an input $x_{p,1}, \dots x_{p,n}$ is given to the network and an output $\hat{y}_{p,1}, \dots, \hat{y}_{p,m}$ is calculated;

weight update where the error between $\hat{y}_{p,1}, \dots, \hat{y}_{p,m}$ and the correct output $y_{p,1}, \dots, y_{p,m}$ is calculated. This error is back propagated through all the layers and a correction Δw_i is calculated for each weight w_i of the network, in order to minimize the error surface in the space of the weights.

In detail, to calculate the weights $\boldsymbol{w}^{(p+1)}$ for the next phase $p+1$, it is sufficient to determine the gradient of the error surface in the current point $\boldsymbol{w}^{(t)}$ in order to apply an optimization method like Stochastic Gradient Descend (SGD). We discuss SGD in detail below.

Loss

The goal of the lerning algorithm is to learn a function f such that predictions $\hat{\boldsymbol{y}} = f(\boldsymbol{x})$ over the training set are accurate respect to the correct labels $\boldsymbol{y}$. The *loss* function measures the prediction error. Formally, given the true expected output $\boldsymbol{y}$, a loss function $\mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y})$ assigns a scalar to a predicted output $\hat{\boldsymbol{y}}$. The loss function should be lower bounded with the minimum value attained only for cases where the prediction is correct. The parameters of the learned function, i.e. the weights $\boldsymbol{w}$, are set in order to minimize the loss over the training examples. Given a labeled training set $\mathcal{D} = (\boldsymbol{x}_{1:n}, \boldsymbol{y}_{1:n})$ and a parameterized model $f(\boldsymbol{x}; \theta)$, the goal of the training algorithm is then to set the values of the parameters θ such that the value of the loss is minimized:

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f(\boldsymbol{x}_i; \theta), \boldsymbol{y}_i). \tag{2.4}$$

The parameters θ represent the set of all the weights w of the ANN. We proceed to describe common loss functions.

Hinge loss

Hinge loss (also known as margin loss) is used in binary classification problems, when the classifier output $\tilde{y}$ is a single scalar and $y \in \{+1, -1\}$. The classification

rule is $\hat{y} = sign(\tilde{y})$ and the classification is considered correct if $y \cdot \tilde{y} > 0$. The hinge loss is defined as:

$$\mathcal{L}_{hinge}(\tilde{y}, y) = \max(0, 1 - y \cdot \tilde{y}).$$

It is 0 when $\tilde{y}$ and y share the same sign and $|\tilde{y}| \geq 1$, otherwise the loss is linear. It attempts to achieve a correct classification with a margin of at least 1.

Hinge loss can be also extended to multi-class settings, where $\hat{\boldsymbol{y}} = \hat{y}_1, \ldots, \hat{y}_n$ are the classifier's output and $\boldsymbol{y}$ the one-hot vector of correct output classes. The classification rule is defined as selecting the class with the highest score $\arg\max_i \hat{\boldsymbol{y}}_i$. Denoting by $t = \arg\max_i y_i$ the correct class index, and by $k = \arg\max_{i \neq t} \hat{y}_i$ the highest scoring class such that $k \neq t$, the multi-class hinge loss is defined as:

$$\mathcal{L}_{hinge}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \max(0, 1 - (\hat{y}_t - \hat{y}_k)).$$

It attempts to score the correct class above all other classes with a margin of at least 1.

Log loss

The log loss is a common variation of the hinge loss, it can be seen as a soft version with an infinite margin [31]. It is defined as:

$$\mathcal{L}_{log}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \log(1 + \exp(-(\hat{\boldsymbol{y}}_t - \hat{\boldsymbol{y}}_k)).$$

Binary cross-entropy loss

The binary cross-entropy loss, also called logistic loss, is used in binary classification with conditional probability outputs. We have a set of two target classes labeled with $y \in \{0, 1\}$. The classifier's output $\tilde{y}$ is transformed using the sigmoid (also logistic) function $\sigma(x) = 1/(1 + e^{-x})$ to the range $[0, 1]$. It can be interpreted as the conditional probability $\hat{y} = \sigma(\tilde{y}) = \mathbb{P}(y = 1|\boldsymbol{x})$. The prediction rule is:

$$\begin{cases} 0 & \hat{y} < 0.5, \\ 1 & \hat{y} \geq 0.5. \end{cases}$$

The network is trained to maximize the log conditional probability for each training example $(\boldsymbol{x}, y)$. The logistic loss is defined as:

$$\mathcal{L}_{logistic}(\hat{y}, y) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y}).$$

While the hinge loss is preferred when we require a hard decision rule, the binary cross-entropy is useful when we want the network to produce class conditional probability.

Categorical cross-entropy loss

The categorical cross-entropy, also called negative log likelihood, is used when a probabilistic interpretation of the scores is desired. Let $\boldsymbol{y} = y_1, \ldots, y_n$ be a vector representing the true multinomial distribution over the labels $1, \ldots, n$, and let $\hat{\boldsymbol{y}} = \hat{y}_1, \ldots, \hat{y}_n$ be the classifier's output transformed by a *softmax* function:

$$softmax(\boldsymbol{x})_i = \frac{e^{x_i}}{\sum_j e^x_j}.$$

$\hat{\boldsymbol{y}}$ represents the class membership conditional distribution $\hat{y}_i = \mathbb{P}(y = i|\boldsymbol{x})$. The categorical cross-entropy loss measures the dissimilarity between the true label distribution $\boldsymbol{y}$ and the predicted label distribution $\hat{\boldsymbol{y}}$. It is defined:

$$\mathcal{L}_{cross-entropy}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\sum_i y_i \log(\hat{y}_i).$$

For hard classification problems in which each training example has a single correct class assignment, $\boldsymbol{y}$ is the one-hot vector of the true class. In such cases the cross-entropy loss can be simplified to:

$$\mathcal{L}_{cross-entropy}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\log(\hat{\boldsymbol{y}}_t),$$

where t is the correct class.

Regularization

The attempt to minimize the loss with (2.4) may result in overfitting the training data, i.e. the model loses the capability to generalize to new data and the loss evaluates poorly on new data that is not present in $\mathcal{D}$. To counter that, we often pose soft restrictions on the form of the solution. This is done using a *regularization* function $R(\theta)$ over the parameters returning a scalar that reflect their *complexity*. This is equivalent to introduce a restriction over the hypothesis space, as seen in section 2.1. We want to keep the model complexity low, Hence we add the regularizer to (2.4) and we let the optimization problem to balance between low loss and low complexity:

$$\hat{\theta} = \arg\min_\theta \left\{ \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f(\boldsymbol{x}_i; \theta), \boldsymbol{y}_i) + \lambda R(\theta) \right\}. \tag{2.5}$$

Different combinations of loss function and regularization criteria result in different learning algorithms with different inductive biases.

Essentially, regularizers work penalizing high values for ANN weights, thus avoiding that the network concentrates only on few features. In (2.5), R is applied to all the parameters, in practice it can be applied on the single layers.

We proceed to describe common regularization functions.

L_2 regularization

In L_2 regularization, also called *gaussian prior* or *weight decay*, R takes the form of the squared L_2 norm of the parameters:

$$R_{L_2}(\boldsymbol{W}) = ||\boldsymbol{W}||_2^2 = \sum_{i,j}(\boldsymbol{W}_{i,j})^2.$$

L_2 regularization penalizes high parameters, but when their values become near to 0 their effect is negligible.

L_1 regularization

In L_1 regularization, also called *sparse prior* or *lasso*, R takes the form of the L_1 norm of the parameters:

$$R_{L_1}(\boldsymbol{W}) = ||\boldsymbol{W}||_1 = \sum_{i,j}|\boldsymbol{W}_{i,j}|.$$

In contrast to L_2, L_1 regularization penalizes uniformly low and high parameters. This has the effect of encouraging a sparse solution [46].

Elastic-net

The elastic-net regularizer [56] combines both L_1 and L_2 regularization:

$$R_{elastic-net}(\boldsymbol{W}) = \lambda_1 R_{L_1}(\boldsymbol{W}) + \lambda_2 R_{L_2}(\boldsymbol{W}).$$

Dropout

Similarly to other regularization techniques, *dropout* method [23] is designed to prevent the network from learning on specific weights. It works by randomly setting to 0 the output of randomly chosen neurons for each sample. The connections between the dropout technique and other regularizations are established [50].

Stochastic Gradient Descend (SGD)

To successfully train a model, it is necessary to solve the optimization problem (2.4). A common solution is to use gradient based methods. Gradient based methods work by estimating the error computing the loss $\mathcal{L}$ over the training set and then calculating the gradients of the error respect to the parameters θ. Once the gradient is computed, the values of the weights are moved in the opposite direction of the gradient.

Stochastic Gradient Descend (SGD) is a general optimization algorithm that is profitably used in training ANN [3, 30]. The hyperparameter ν_t is the learning rate. It can either be fixed throughout the learning process, or decay as a function of the

Input: Parametrized function $f(\boldsymbol{x}; \theta)$
Training set $\mathcal{D}(\boldsymbol{x}_{1:n}, \boldsymbol{y}_{1:n})$
while *stopping criteria not met* **do**
  Sample training example $\boldsymbol{x}_i, \boldsymbol{y}_i$;
  Compute $\mathcal{L}(f(\boldsymbol{x}_i; \theta), \boldsymbol{y}_i)$;
  $\hat{\boldsymbol{g}} \rightarrow$ gradients of $\mathcal{L}(f(\boldsymbol{x}_i; \theta), \boldsymbol{y}_i)$ w.r.t. θ;
  $\theta \rightarrow \theta - \nu_t \hat{\boldsymbol{g}}$
end

Algorithm 1: Stochastic Gradient Descend (SGD).

time step t. Adam is a widely used algorithm that adaptively change the learning rate [28]. The error calculated in algorithm 1 is based on only one example. This may result in an inaccurate gradient calculation. A common way to reduce the noise of this inaccuracy is to estimate the error and the gradients on a sample of m examples. This is called minibatch SGD.

Recurrent Neural Network (RNN)

RNNs are kinds of ANNs specialized for sequences. They exhibit an internal state $\boldsymbol{h}$ that changes during the training and that recursively depends on the state of the previous phase. Precisely we have eq. (2.6):

$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}), \tag{2.6}$$

where $\boldsymbol{h}$ is the state vector, $\boldsymbol{x}$ is the input vector and $\boldsymbol{\theta}$ are the parameters of the state-function f. The index t indicates the iteration number and can be interpreted as a discrete time or, more in general, as the progressive number of the sequence that is presented as input.

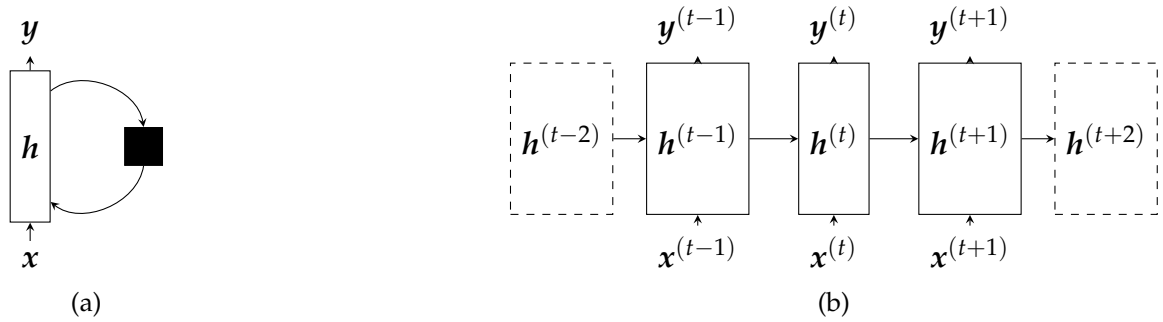


Figure 2.3: RNN, folded (a) and unfolded (b) models.

In order to express a compact visualization of RNNs, it is possible to use the computational graph in fig. 2.3a where the black box is a delay of one iteration. An extract of the unfolding of the computation graph is shown in fig. 2.3b.

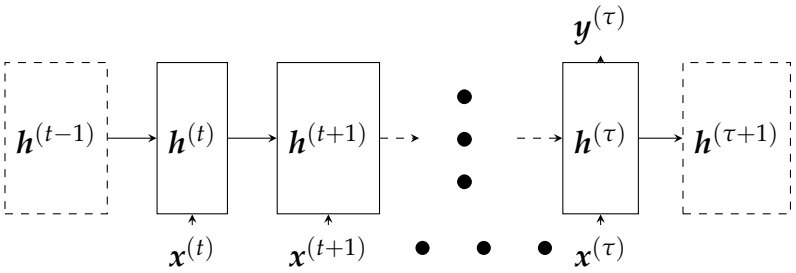


Figure 2.4: RNN with single output after a sequence of inputs.

The model in fig. 2.3 provides an output $\boldsymbol{y}^{(t)}$ for every input $\boldsymbol{x}^{(t)}$. An alternative is the model of fig. 2.4 where an output $\boldsymbol{y}^{(\tau)}$ is provided only after a sequence $\boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(\tau)}$ of inputs.

In order to train a RNN, it is sufficient to apply the backpropagation to the entire unfolded model.

Long Short-Term Memory (LSTM)

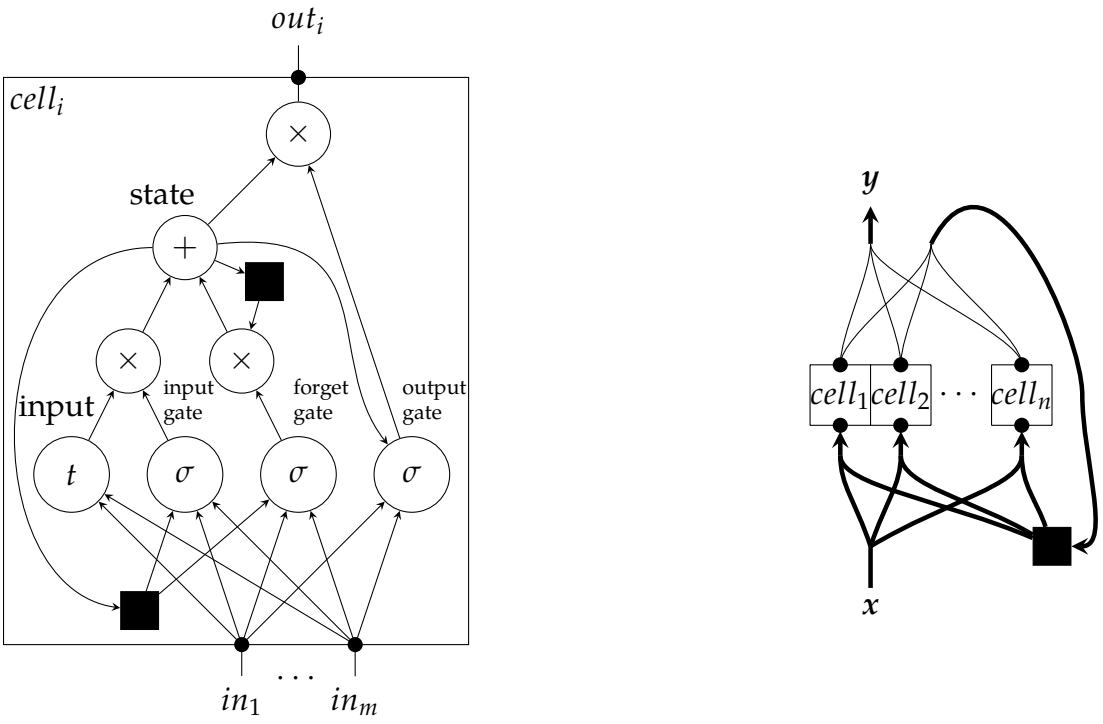
The major drawback of RNNs is the *vanishing gradient* problem during the backpropagation. The gradient for long-term associations, propagated through many stages, tends to become zero. A possible solution for this problem is LSTM [24].

The LSTM model is structured as in fig. 2.5. It is equivalent to a generic RNN where the hidden state $\boldsymbol{h}$ is a layer of *memory cells*.

In detail, a single memory cell (fig. 2.5a) has four Artificial Neuron (AN). One is labelled *input* and processes the cell's inputs into an internal state. The other three ANs are labelled as *gates* and process the cell's inputs together with the previous-phase state in order to decide:

- how much of the current input must be learned (*input gate*);
- how much of the previous-phase state must be forgotten (*forget gate*);
- how much of the state constitutes the output (*output gate*).

The memory-cells layer (fig. 2.5b) is composed of a number of cells equal to the dimension of the output. Each cell calculates one dimension of $\boldsymbol{y}$. The input $\boldsymbol{x}$ is copied for every cell and, together with the previous-phase output, constitutes the input $in_1, \ldots, in_m$ of the single cells.



(a) (b)

Figure 2.5: LSTM model: detail of memory cell (a), and general scheme (b). The black box is a delay of one iteration.

Gated Recurrent Unit (GRU)

Gated Recurrent Units (GRUs) were introduced by [9] as an alternative to LSTM. GRU is also based on gating mechanism, but it has fewer gates respect to LSTM and the memory is directly exposed to the output. One gate r called *reset* is used to control access to the previous state h_{t-1} and computes a proposed update $\tilde{h}$. The new status h_t is calculated as an interpolation between the previous and the proposed status, where the proportion of the interpolation is controlled by the *update* gate z. Formally, at time t the GRU computes its new state h_t as:

$$h_t = (1 - z_t) \bigodot h_{t-1} + z_t \bigodot \tilde{h}_t,$$

and the proposed new state $\tilde{h}_t$ as:

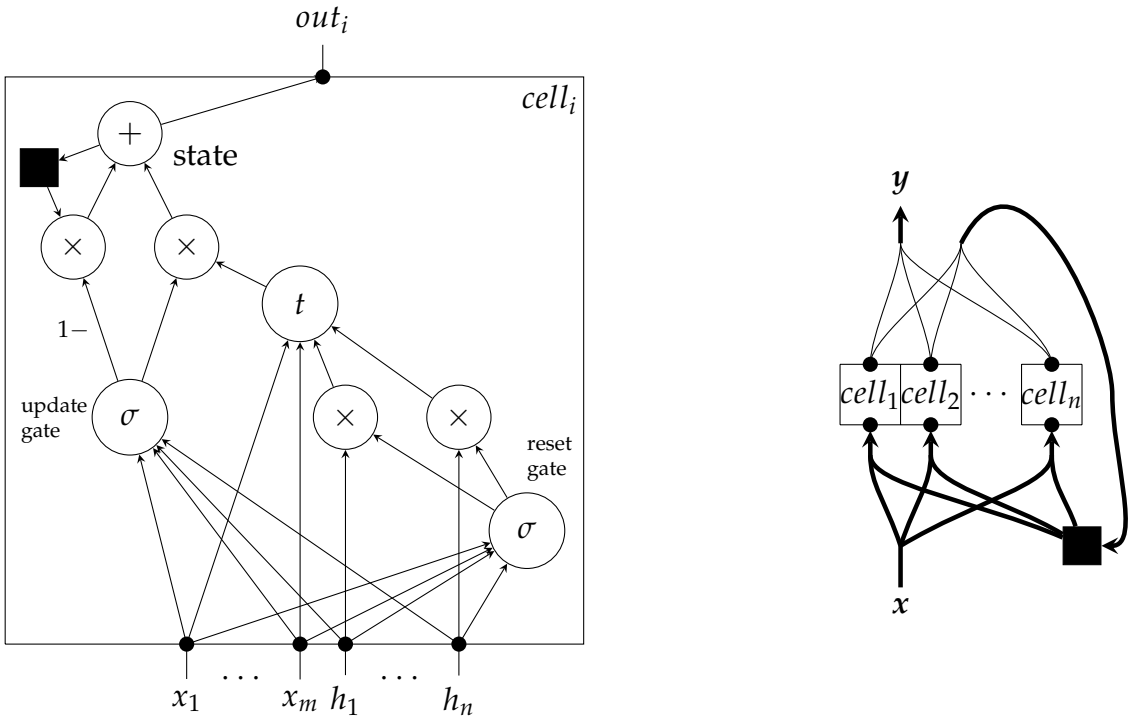
$$\tilde{h}_t = tanh(W_h x_t + r_t \odot (U_h h_{t-1}) + b_h).$$

The update gate z controls how much information from the past should be keep and how much new information should be have. z_t is updated as:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + bz).$$

The reset gate r controls how much of the past state contributes to the candidate. It is calculated as:

$$r_t = \sigma(W_r x_r + U_r h_{t-1} + b_r).$$



(a) (b)

Figure 2.6: GRU model: detail of memory cell (a), and general scheme (b). The black box is a delay of one iteration.

The choice between LSTM and GRU is context dependent, in certain applications LSTM gives better performances, in other applications GRU does [54].

Chapter 3

Text analysis with deep learning

3.1 NLP

NLP is the field of designing methods that takes natural language data as an input or produces it as an output. Natural language is ambiguous and variable, for instance the sentence "i saw a woman on a hill with a binocular" can means that I had a binocular and with those I saw a woman on a hill, or that a woman was on a hill using the binocular. This ambiguity can also be the source of problems in a specific domain like the one of the medical text, and specific countermeasures may be adopted [10, 55].

Language is *symbolic* and *discrete*, the relationship between different words, i.e. symbols, cannot be inferred from the symbols themselves. It is possible to easily compare concepts that have a continuous representation, e.g. two different colors in an image, while it cannot be done easily with words without using large lookup table or advanced methods. Language is also *compositional*, letters form words, and words form sentences. Language is also *sparse*, the way in which words can be combined to form meanings is practically infinite.

3.2 ANN in NLP

ML approaches are characterized by learning to make predictions based on past observations. ANN approaches work by learning not only to predict, but also to correctly represent data. A common component of ANN applied to NLP is the embedding layer, i.e. a mapping from discrete symbols to continuous vectors in a relatively low dimensional space. This representation allows to transform the isolated symbols into mathematical objects that can be operated on.

The primary model that is used in NLP are RNNs. They are capable of producing a vector that summarizes the entire input sequence. With them, it is possible to abandon the *markov* assumption that was prevalent in NLP for decades, and to

design models that can condition on entire sentences or documents. This capability leads to impressive progress in *language-modeling*, the task of predicting the probability of the next word in a sequence.

Features

The mapping from textual data to real valued vectors that can be used as input for ANN models, is called *features extraction*.

When the focus entity is a word out of context, the main source of information is the letters. We can look at a *lemma*, i.e. dictionary entry of the word, e.g. words such as "booking", "booked", "books" have "book" as their common lemma. This mapping is usually performed using lemma lexicons or morphological analyzers. It is a linguistically defined process and may not work well for forms that are not in the lexicon or for mispellings. A coarser process is called *stemming*, it maps words to shorter words that are not necessarily grammatically valid, e.g. "picture", "pictures", and "pictured" will be stemmed to "pictur". *Lexical resources* are dictionary that are meant to be accessed by machines rather than humans. They typically contain information about words, e.g. there are lexicons that map inflected word forms to their possible morphological analyses, informing that a certain word may be a singular masculine noun or a past perfect verb.

When the focus entity is text, i.e. sentences or paragraphs documents, the features are the count and the order of the letters and words within the text. *Bag of words* is a very common feature extraction procedure. We look at the histogram of the words within the text. We can compute quantities that are directly derived from the words and the letters, such as the length of the sentence. We can also integrate statistics based on external information. When using bag of words, it is common to use Term-Frequency Inverse-Document-Frequency (TF-IDF) weighting [34]. A word w in a document d that is part of a large corpus D of documents is represented by:

$$\frac{\#_d(w)}{\sum_{w' \in d} \#_d(w')} \cdot \log \frac{|D|}{|d \in D : w \in d|},$$

where $\#_d(w)$ is the number of times that w appears in d. Besides words, one may also look at consecutive pairs or triplets of words. These are called *ngrams*. A bag of ngrams representation is much more informative than a bag of words.

When considering a word within a sentence or a document, the features of a word are its position within the sentence and the words or letters surrounding it. It is common to focus on the immediate context of a word by considering a *window* surrounding it (with typical values of 2, 5, and 10 words to each side). We may also be interested in the absolute position of a word inside a sentence, having features such as "the word is the 5th in the sentence" or "the word appears within the first 10 of the sentence".

When considering more than a word within a context, we can also look at the text *distance* between them or the identities of the words that appear between them.

In natural language, sentences have structures beyond the linear order of their words. The structure is not directly observable and is referred to as *syntax*. While it is not observable, it can be inferred from the sentence. Specialized systems exist for the prediction of parts of speech, syntactic trees, semantic roles, discourse relations, and other linguistic properties. These prediction often serve as good features for classification problems.

Different features can also be combined together. Instead of combining them manually, we can provide a set of core features to an ANN model and rely on the training procedure to pick up important combinations of them. Core features can also be learned by ANN, but enough data is needed. The distributional hypothesis of language states that the meaning of a word can be inferred from the contexts in which it is used. By observing co-occurrence patterns of words across a large body of text, it is possible to infer that a word is similar to another word. Many algorithms were derived to make use of this property. They can be categorized into clustering-based methods that assign similar words to the same cluster and represent each word by its cluster membership [37], and embedding-based methods which represent each word as a vector such that similar words (with a similar distribution) have similar vectors [36, 39].

3.3 Text classification

The classic approach is to employ bag-of-words representations of textual documents [34]. In this approach, a document is described by a *set* or a *multiset* of words. Multisets allow one to take into account the number of occurrences of a word in the document. Vector representations of documents are easily derived from bag-of-words. When using unigrams, the dimensionality of each vector equals the size of the vocabulary in use. In the simplest case, the vector x representing a document has Boolean components $x_j = 1$ if and only if term j appears in the document. A slightly more informative representation, derived from multisets, is TF-IDF [34]. In this case $x_j = n_j \log \frac{|D|}{|D_j|}$, where n_j is the number of times term j occurs in the document, $|D|$ is the cardinality of the data set, and D_j is the set of documents containing term j. In those representations, common terms receive a lower weight. An alternative representation suggested in [35] employed within-category frequencies but only yielded modest improvements over TF-IDF.

A more informative representation of documents can be obtained by considering bigrams and trigrams, i.e. pairs or triplets or terms that occur consecutively in a document. These representations are suitable for large data sets and have been commonly employed in other contexts.

Bag-of-words representations (including those employing bigrams or trigrams) enable the application of very simple text classifiers like Naive Bayes (NB) or SVM [12]. However, they suffer two fundamental problems. First, the relative order of terms in the documents is lost, making it impossible to take advantage of the syntactic structure of the sentences. Second, distinct words have an orthogonal representation even when they are semantically close. Moreover, with this representation the vast majority of the dataset, i.e. the unlabeled records, remains unused. As detailed in section 3.4, word vectors can be used to address the second limitation and also allow us to take advantage of unlabeled data, which can be typically obtained in large amounts and with little cost.

3.4 Word Vectors

Many modern approaches to NLP take advantage of vector-space word representations to solve specific tasks such as retrieval, classification, named entity recognition or parsing. The use of word vectors eliminates the need for complex ontologies like WordNet [18], that express various kinds of semantic relations among words (such as synonymy, hypernymy, meronymy, etc). Word vectors are typically constructed in such a way that analogies are directly encoded in vector space. One often cited example of analogy is "king is to man as queen is to woman", which should correspond to the similarity in vector space [36]:

$$x_{king} - x_{man} + x_{woman} \approx x_{queen}.$$

In a similar spirit, one could imagine the following vector space similarity to occur in the oncology domain:

$$x_{glioma} - x_{glia} + x_{connective} \approx x_{fibroma}.$$

Most algorithms for obtaining word vectors are based on co-occurrences in large text corpora. Co-occurrence can be measured either at the word-document level (e.g. using latent semantic analysis) or at the word-word level (e.g. using word2vec [36] or Global Vectors (GloVe) [39]). It is a common practice to take advantage of precompiled libraries of word vectors trained on several billion tokens extracted from various sources such as Wikipedia, the English Gigaword 5, Common Crawl, or Twitter. These libraries are generally conceived for general purpose applications and are only available for the English language. Since our cancer registry textual data is written in Italian and employs a very specific domain terminology, we retrained word vectors from the almost 1.5 millions unlabeled records described in Section 4.1. For this purpose, we trained GloVe [39]. An excerpt of the trained dataset is visible in fig. 3.1.

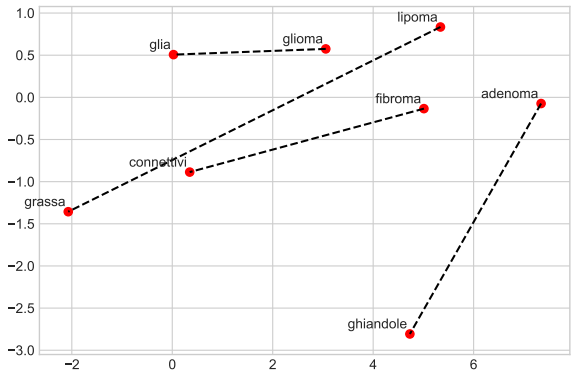


Figure 3.1: Extract from constructed vector space. The Italian labels are *grassa* for *fat*, *connettivi* for *connectives*, *ghiandole* for *glands*.

The training process involves the construction of the $n \times n$ triangular matrix C of words' co-occurrence, where n is the number of unique words $w_1, \ldots, w_n$ inside the text (possibly with the exclusion of the most and the least frequent terms). In order to do this, a window of size ω slides through the text. After the construction of C, GloVe uses the information contained in it to train a model that produces vector of specific dimension ν. ω, ν, and the number of training iterations η are the hyperparameters of the method.

3.5 Attention models

Conditioned generation with attention is a powerful architecture. They find the main application in the context of sequence-to-sequence generation.

The first results with attentive models where due to Bahdanau et al. [2], who used an architecture based on two parts. The task of the first part is to encode the input using an RNN to create hidden representations of the sequence. Contrarily to previous approaches, the sequence is not encoded in a single representation, but each word of the input has its representation. The second part generates the output sequence, with a generative RNN, using a weighted average of the input representation to condition the generation. The weights are calculated by the attention mechanism, based on the values of the hidden representation of the input and the state of the output generating RNN. The attention mechanism is trained together with the rest of the model. Luong et al. [33] explored variation on the attention mechanism leading to some improvement.

3.6 Hierarchical models

Recent works on attention employ hierarchical structures of the models. Yang et al. [52] proposed a model called Hierarchical Attention Network (HAN) that combines two characteristics: a hierarchical structure that reflects the hierarchical structure of the text and an attention mechanism applied to each one of the two methods. Their hypothesis was that a better text representation can be obtained incorporating the document structure in the model. Their model is structured in two levels: a sentence encoder that uses a bidirectional GRU layer to extract hidden representations of each word, followed by an attention mechanism that aggregates them; a document encoder that repeats the same process obtaining document representations from the sentences representations. The attention mechanism calculates the importance of a word as the similarity between a hidden word representation and a learned context vector.

Gao et al. [21] applied HAN [52] in the context of cancer pathology reports. They implemented two classification tasks on 942 reports: one with 12 main site ICD-O codes and the other with 4 histological grade. They demonstrated that in their tasks HAN outperforms other conventional machine learning and deep learning approaches.

Wang et al. [51] organized ngrams in a hierarchical structure, and used a tree-structured LSTM to achieve a self-explaining text classification model. They evaluated the model on a public medical text classification dataset.

3.7 BERT

BERT [16] is a recent model that represents the state of the art in many NLP related tasks [7, 25, 32, 48]. It is a bi-directional pre-training model supported by the Transformer Encoder [49]. This model is attention-based and learns context-dependent word representation on large unlabeled corpora, and then the model is fine tuned end to end on specific labeled tasks. During pre-training, the model is trained on unlabeled data over two different tasks. In Masked Language Model (MLM) some tokens are masked and the model is trained to predict those tokens based on the context. In Next Sentence Prediction (NSP) the model is trained to understand the relationship between sentences predicting if two sentences are actually consecutive or if they where randomly replaced (with 50% probability). After the pre-training, the model is fine-tuned to the specific task.

Chapter 4

Materials and Methods

4.1 Dataset

We collected a set of 1 592 385 anatomopathological exam results from Tuscany region Registro Tumori della Toscana, Tumor Registry of Tuscany (RTT) in the period 2004-2013. About 6% of these records refer to a positive tumor diagnosis and have topological and morphological ICD-O3 labels, determined by tumor registry experts. Other reports are associated with non-cancerous tissues and with unlabeled records. When multiple pathological records for the same patient existed for the same tumor, experts selected the most informative report in order to assign the ICD-O3 code to that tumor case, leaving a set of 94 524 labeled tumor cases.

The histological exam records consist of three free-text fields (not all of them always filled-in) reporting tissue macroscopy, diagnosis, and, in some cases, the patient's anamnesis. We found that field semantics was not always used consistently and that the amount of provided details varied significantly from extremely synthetic to very detailed assessments of the morphology and the diagnosis. Field length ranged from 0 to 1 368 words, with first quartile, median and third quartile respectively 34, 62, 134. For these reasons, we merged the three text fields into a single text document. We did not apply any conflation (except for case normalization) and we kept punctuation.

We also collected the remaining unlabeled part of the dataset to train unsupervised methods.

For some of our experiments, we further split the records in sentences for the use on hierarchical models. For this purpose we employed the *spaCy* sentence segmentation[1]. We note that the distribution of our documents in fig. 4.1 is different respect to the one in [21]. The distributions of words per sentence are similar, but our documents contains on average less sentences per document.

[1] `https://spacy.io/`

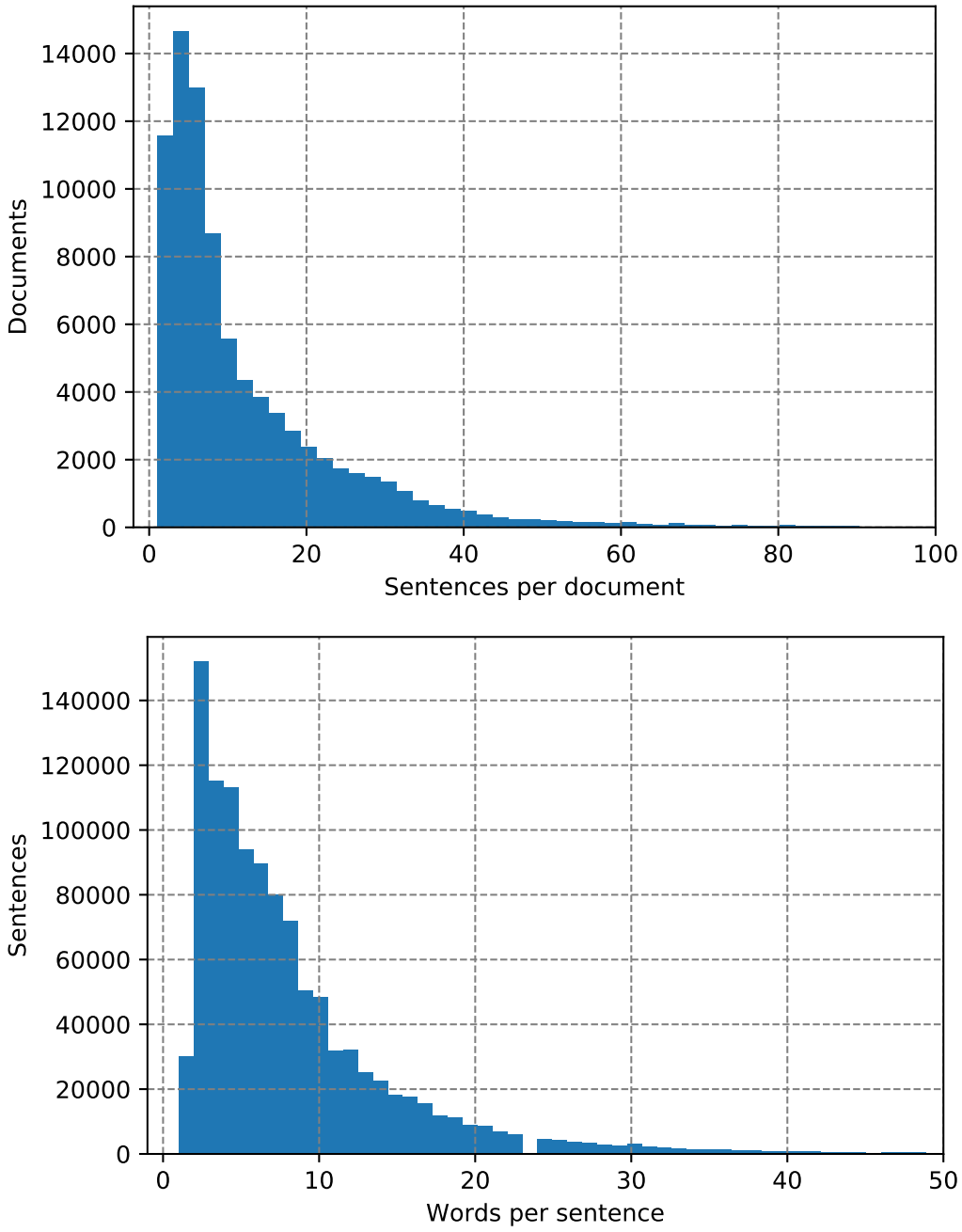


Figure 4.1: Distribution of the number of sentences per document (top) and the number of words per sentence (bottom).

Preparation

Originally, the data comes in two tables. The first one is composed of a list of neoplasm records, for each one are specified: a series of administrative variables, e.g. the date of incidence, the hospital; ICD-O codes, both first and third editions, for topography and morphology; other clinical variables, e.g. *Gleason*, *Clark*, *Dukes* scores. ICD-O codes inside neoplasm records are assigned by RTT personnel, thus they can be considered reliable and used as ground truth for the learning models.

Furthermore, there is a histology table containing records resulting from microscopy exams. Each record contains three free-text fields: one for the macroscopy, one for the diagnosis, and one for other information.

The neoplasm and histology tables can be joined using a neoplasm identifier. Thus connecting the text fields with the true ICD-O codes.

There are neoplasm cases without a related histology exam because RTT uses also other sources to collect tumor cases, e.g. Hospital Discharge Registers (HDRs) and death certificates. There are also histology exams without a related neoplasm case because not always a histology results in a tumor case.

To train the unsupervised methods, we kept all the records from the histology table that cannot be joined with the neoplasm table.

4.2 Models

In total, we created fourteen models:

U-SVM uses SVM trained on TF-IDF representations of text using unigrams;

B-SVM uses SVM trained on TF-IDF representations using unigrams and bigrams;

B-XGB uses gradient boosted decision trees [8] on TF-IDF representations using unigrams and bigrams

B-LSTM uses LSTM layers trained on TF-IDF representations using bigrams;

G-CRNN uses mixed convolutional and LSTM layers trained on GloVe representations;

G-LSTM uses LSTM trained on GloVe representations;

G-GRU uses GRU layers trained on GloVe representations;

G-MAX uses GRU layers with max pooling trained on GloVe representations;

G-ATT uses GRU layers with attention trained on GloVe representations;

G-MAXi uses GRU layers with max pooling, in an interpretable setting, trained on GloVe representations;

G-ATTi uses GRU layers with attention, in an interpretable setting, trained on GloVe representations;

G-MAXh uses hierarchical GRU layers with max pooling trained on GloVe representations;

G-ATTh uses hierarchical GRU layers with attention trained on GloVe representations;

BERT is the model in [16] pretrained on our unlabeled data and fine tuned with our labeled data.

In models ***U-SVM*** and ***B-SVM***, we used a TF-IDF representation ignoring terms present in less than 3 documents or in more than 30% of the documents.

We trained all the models except SVM minimizing the *categorical crossentropy*.

In ***B-LSTM*** we used a word embedding of 30. The embedding layer corresponds to a one-hot representation of the input followed by a dense layer of 30 neurons.

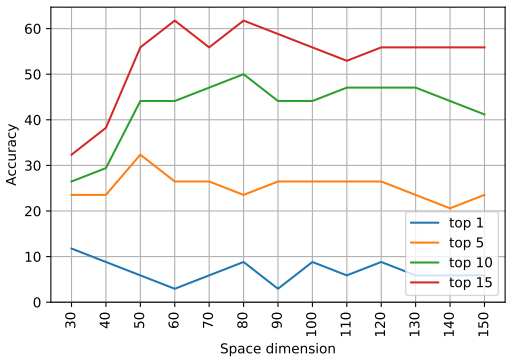


Figure 4.2: Accuracy top 1,5,10 and 15 for an intrinsic test with varying word vector dimension.

In ***G-LSTM*** and ***G-CRNN*** we used the word vector representation explained in section 3.4. We trained GloVe with a window dimension of 15, in 50 iterations to produce representations in dimension 60. To decide these parameters, we developed intrinsic tests collecting quadruples like:

$$(melanoma,\ cute,\ duttale,\ mammella)$$

translated:

$$(melanoma,\ skin,\ ductal,\ breast)$$

in order to verify if $\boldsymbol{x}_{breast}$ is near $\boldsymbol{x}_{skin} - \boldsymbol{x}_{melanoma} + \boldsymbol{x}_{ductal}$. Then we proceeded to confront the different parameters as in fig. 4.2.

In order to accelerate the computation for the models ***B-LSTM***, ***G-LSTM*** and ***G-CRNN***, we cut the length of text to 200. The 87% of records have less than 200 words.

B-LSTM is the ANN in fig. 4.3. It is composed of two layers of 150 bidirectional LSTM cells, followed by an average pooling of the sequences, followed by a dense Rectified Linear Unit (ReLU) layer, followed by a dense *softmax* layer. The number of ANs of the last two layers is equal to the number of classes for each task.

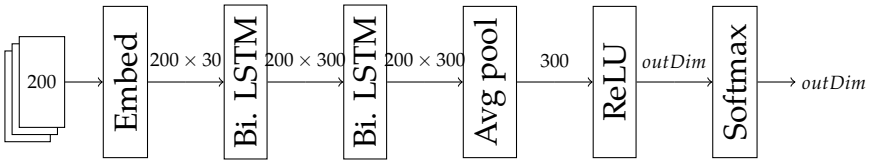


Figure 4.3: Scheme for ***B-LSTM*** model.

G-CRNN, in fig. 4.4, is composed of a convolutional filter of size two, followed by a bidirectional LSTM layer of 150 cells, followed by an average pooling of the sequences, followed by a dense ReLU, followed by a dense *softmax* layer.

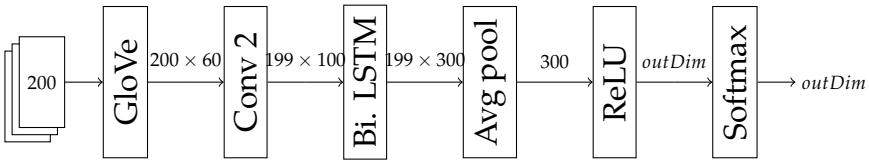


Figure 4.4: Scheme for ***G-CRNN*** model.

G-LSTM, in fig. 4.5, is composed of two bidirectional LSTM layer of 150 cells, followed by an average pooling of the sequences, followed by a dense ReLU, followed by a dense *softmax* layer.

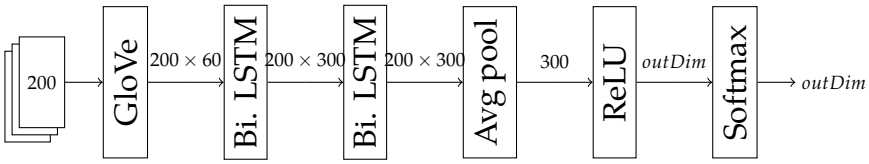


Figure 4.5: Scheme for ***G-LSTM*** model.

G-GRU is a stacked bidirectional GRU model.

The other models where trained on different hyperparameters configurations whose range is explained in section 5.3. We proceed to explain in detail their structure.

G-MAX, G-ATT

In our setting, a dataset $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}$ consists of variable length sequence vectors $\boldsymbol{x}^{(i)}$. For $t = 1, \ldots, T^{(i)}$, $x_t^{(i)}$ is the t-th word in the i-th document and $y^{(i)} \in \{1, \ldots, K\}$ is the associated target class. To simplify the notation in the subsequent text, the superscripts are not used unless necessary. Sequences are denoted in boldface. The RNN-based sequence classifiers used in this work compute their predictions $f(\boldsymbol{x})$ as follows:

$$e_t = E(x_t; \theta^e), \tag{4.1}$$

$$h_t^f = F(e_t, h_{t-1}^f; \theta^f), \tag{4.2}$$

$$h_t^r = R(e_t, h_{t+1}^r; \theta^r), \tag{4.3}$$

$$u_t = G(h_t; \theta^h), \tag{4.4}$$

$$\phi = A(\boldsymbol{u}; \theta^a), \tag{4.5}$$

$$f(\boldsymbol{x}) = g(\phi; \theta^c). \tag{4.6}$$

E is an embedding function mapping words into p-dimensional real vectors where embedding parameters θ^e can be either pretrained and adjustable or fixed, see Section 3.4 below. Functions F and R correspond to (forward and reverse) dynamics that can be described in terms of several (possibly layered) recurrent cells. Each vector h_t, the concatenation of h_t^f and h_t^r, can be interpreted as latent representations of the information contained at position t in the document. G is an additional MLP mapping each latent vector into a vector u_t that can be seen as contextualized representation of the word at position t. A is an aggregation function that creates a single d-dimensional representation vector for the entire sequence and g is a softmax layer. Possible choices for the aggregator function include:

- $\phi = (h_T^f, h_1^r)$ (in this case G is the identity function), which extracts the extreme latent representations; in principle, these may be sufficient since they depend on the whole sequence due to bidirectional dynamics. However, note that this approach may require long-term dependencies to be effectively learned;
- $\phi = \sum_t a_t(\boldsymbol{u}; \theta^a) u_t$, using an attention mechanism as in [52]. In this case, (scalar) attention weights are computed as

$$c_t = C(\boldsymbol{u}; \theta^a),$$

$$a_t(\boldsymbol{u}; \theta^a) = \frac{e^{\langle c, c_t \rangle}}{\sum_i e^{\langle c, c_i \rangle}},$$

  where C is a single layer that maps the representation u_t of the word to a hidden representation c_t. Then, the importance of the word is measured as a similarity with a context vector c that is learned with the model and can be seen

as an embedded representation of an high level query as in memory networks [43];

- $\phi_j = \max_t u_{j,t}$. This approach is closely related to the bag-layer proposed in the context of multi-multi-instance learning [45]. In this case, each "feature" ϕ_j will be positive if at least one of $u_{j,1}, \ldots, u_{j,T}$ is positive. The resulting classifier will find it easy to create decision rules predicting a document as belonging to a certain class if a given set of contextualized word representations are present and another given set of contextualized word representations are absent in the sequence. Note that this aggregator can also be interpreted as a kind of hard attention mechanism where attention concentrates completely on a single time step but the attended time step is different for each feature ϕ_j. As detailed in Section 5.2, a new model interpretation strategy can be derived when using this aggregator.

The parameters θ^f, θ^r, θ^h, and θ^a (if present) are determined by minimizing a loss function $\mathcal{L}$ (categorical cross-entropy in our case) on training data:

$$\hat{\theta} = \arg\min_{\theta} \sum_{(\boldsymbol{x},y)\in\mathcal{D}} \mathcal{L}\left(y, f(\boldsymbol{x})\right), \tag{4.7}$$

where $\theta = \theta^f \cup \theta^r \cup \theta^h \cup \theta^a$.

We call ***G-MAX*** the model $f(\boldsymbol{x})$ when setting the aggregator function A equal to the max pooling $\max_t u_t$. When we use the attention $\sum_t a_t(\boldsymbol{u};\theta^a)u_t$ instead, we call the model ***G-ATT***. When the aggregating function is equal to $\phi = (h^f_T, h^r_1)$ the model represents a plain bidirectional RNN and we call it ***G-GRU***.

G-MAXi, G-ATTi

The model is flexible enough to gain interpretability under certain assumptions. If we remove the last layer in (4.6), the size of last layer in (4.4) needs to be equal to the output size. The model in this configuration compute their predictions $f(\boldsymbol{x})$ as follows:

$$e_t = E(x_t;\theta^e), \tag{4.8}$$
$$h^f_t = F(e_t, h^f_{t-1};\theta^f), \tag{4.9}$$
$$h^r_t = R(e_t, h^r_{t+1};\theta^r), \tag{4.10}$$
$$u_t = G(h_t;\theta^h), \tag{4.11}$$
$$f(\boldsymbol{x}) = A(\boldsymbol{u};\theta^a), \tag{4.12}$$

where E, F, R, G and A are defined as in section 4.2.

We hypothesized that, in this case, the values of u_t (or the weighted values $a_t(\boldsymbol{u}; \sigma^a)u_t$, if we use the attention as aggregator) collect information on the importance of the area around x_t for the purpose of classification task. This information can be used to interpret the model decision. We call ***G-MAXi*** this interpretable setting when using the max aggregator, and ***G-ATTi*** when using the attention.

G-MAXh, G-ATTh

We extend the plain model of section 4.2 with a hierarchical setting, similarly to other works [52]. In this setting our dataset $\mathcal{D} = \{\boldsymbol{x}^{(i)}, y^{(i)}\}$ consists of variable length sequence of sequence vectors $\boldsymbol{x}^{(i)}$, where, for $s = 1, \ldots, S^{(i)}$ and $t = 1, \ldots, T^{(i,s)}$, $x^{(i)}_{s,t}$ is the t-th word of the s-th sentence in the i-th document, and $y^{(i)} \in \{1, \ldots, K\}$ is the associated target class. The prediction $f(\boldsymbol{x})$ is calculated:

$$e_{s,t} = E(x_{s,t}; \theta^e), \tag{4.13}$$
$$h^f_{s,t} = F(e_{s,t}, h^f_{s,t-1}; \theta^f), \tag{4.14}$$
$$h^r_{s,t} = R(e_{s,t}, h^r_{s,t+1}; \theta^r), \tag{4.15}$$
$$u_{s,t} = G(h_{s,t}; \theta^h), \tag{4.16}$$
$$\phi_s = A(\boldsymbol{u}_s; \theta^a), \tag{4.17}$$
$$\bar{h}^f_s = \bar{F}(\phi_s, \bar{h}^f_{s-1}; \bar{\theta}^f), \tag{4.18}$$
$$\bar{h}^r_s = \bar{R}(\phi_s, \bar{h}^r_{s+1}; \bar{\theta}^r), \tag{4.19}$$
$$\bar{\phi} = \bar{A}(\bar{\boldsymbol{h}}; \bar{\theta}^a), \tag{4.20}$$
$$f(\boldsymbol{x}) = g(\bar{\phi}; \theta^c). \tag{4.21}$$

As in the plain model, E is an embedding function, F and R correspond to forward and reverse dynamics that process word representations, $h_{s,t} = h^f_{s,t} \oplus h^r_{s,t}$ is the latent representation of the information contained at position t of the s-th sentence, $u_{s,t}$ is the contextualized representation of the word at position t of the s-th sentence, and A is an aggregation function that creates a single representation for the sentence. Furthermore, $\bar{F}$ and $\bar{R}$ correspond to forward and reverse dynamics that process sentence representations, and $\bar{A}$ is the aggregation function that creates a single representation for the entire document. $\bar{h}_s = \bar{h}^f_s \oplus \bar{h}^r_s$ can be interpreted as the latent representation of the information contained in the sentence s for the document.

We call ***G-MAXh*** and ***G-ATTh*** the hierarchical model when we use respectively the max pooling and the attention.

Chapter 5

Experiments

With the experiments we want to show the feasibility of deep learning methods and, more generally, machine learning methods applied to large scale histological records. Moreover, we want to compare classical bag-of-word techniques with recent deep learning methods. We also want to assess the effects of leveraging large corpora of unlabeled text that comes from the same distribution, in the context of text classification. Finally, we want to check if attention-based methods determine an improvement, especially when used in a hierarchic way like in previous works. We also want to show some practical use cases and evaluate the interpretation of deep learning models.

In appendix A, we describe some useful metrics that can be used to assess the classifier behavior and also to use the classifiers in different use cases.

In section 5.1, we organize a comparative study between SVM and deep learning techniques using both bag-of-words and word vectors. We trained the models ***U-SVM***, ***B-SVM***, ***B-LSTM***, ***G-CRNN***, and ***G-LSTM***in 10-fold cross validation. We calculated different metrics for all the models and we summarized the average and standard deviation along the folds. In these experiments we can appreciate the improvements of using word vectors. Furthermore, we can notice that by using SVM, either on unigrams and bigrams, we already achieved good results.

Related to the motivations in section 1.4, with these experiments we want to prove Q1. Also we want to answer to Q3 by comparing LSTM with SVM with both GloVe and bag-of-word features. With the training of word vectors we also want to answer to Q5.

In section 5.2 we want to investigate the potentialities of the attention based models and the simpler max aggregation. We designed an artificial experiment where the two different kinds of aggregation are compared. Moreover, we also used this artificial dataset to preliminarily investigate the interpretability potentialities of ***G-MAXi*** and ***G-ATTi***. With these experiments we want to show that the attention model is not always better than a simpler max aggregation.

Regarding to the questions in section 1.4, these artificial experiments are a preliminary answer to Q4 and Q6 because we confront attention-based techniques with simpler aggregation techniques. Also in these experiments we start to study the interpretability of the models.

In section 5.3, we changed the setting, instead of doing a 10-fold cross validation we adopt a temporal split. This decision was made to reflect a probable practical employment of the classifier, where it is being trained on the past data and evaluated on future data. In these experiments we are interested in evaluating the effects of the attention models, comparing them with a simpler form constituted by the max aggregation of ***G-MAX***. We focused on GRU because preliminary experiments did not show an improvement of using LSTM. With these experiments, we want to show that in this context the adoption of hierarchical models is not beneficial. Moreover, we want to evaluate how the interpretable models work respect to the others.

Regarding the questions in section 1.4, these experiments answer to Q1 (with different settings respect to the experiments in section 5.1). We answer to Q2 and Q4 because we apply attention and hierarchical models to the cancer pathology reports. We also want to answer to Q6 because we implement interpretable models.

5.1 Bag-of-words VS word vectors, SVM VS deep learning

The focus of this section is to asses the improvement of employing word vectors respect to the use of bag-of-words and to evaluate the employment of deep learning techniques.

We realized four multiclass classifiers for the histological exams. Calling X the distribution of texts, Y^s of site, Y^f of full site (site plus subsite), Y^t of type, and Y^b of behavior:

Main-site estimates $P(Y^s|X)$;

Full-site estimates $P(Y^f|X)$;

Type estimates $P(Y^t|X)$;

Behavior estimates $P(Y^b|X)$.

The tasks have a variable number of represented classes, summarized in table 5.1. Moreover, data is not balanced. As visible in fig. 5.1, some classes are common, while many are rare.

The frequency of words in the text follows the typical Zipf's distribution with few words that cover the majority of text and a long tail of infrequent words.

Table 5.1: Number of classes for different tasks.

task	classes
Main-site	70
Full-site	284
Type	434
Behavior	5

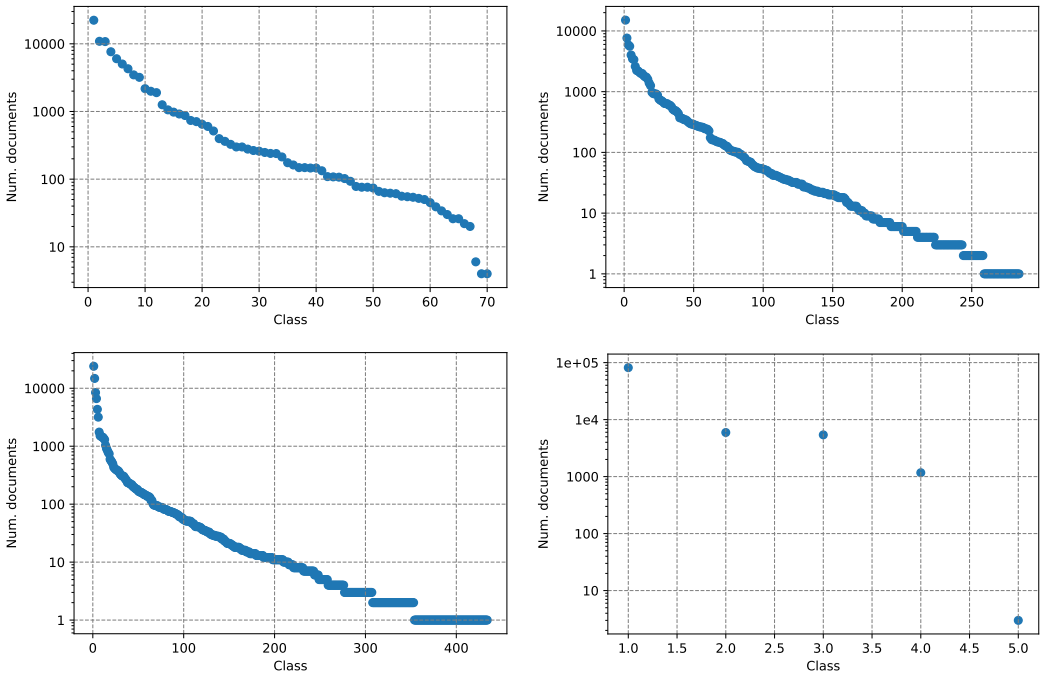


Figure 5.1: Documents in classes (ordered by frequency).

Experiments

We used leave-one-out ten-folds cross validation to take advantage of all the available data. The whole pathological-record dataset was split in ten equal parts called folds. To preserve labels distribution, this was performed in a stratified way (all the folds have the same proportions of labels). Afterwards, each model was trained ten times, using one fold at a time as test dataset and the remaining as training. For each metric we summarized the average and the standard deviation among the runs. Regarding the curves we calculated the cumulative for each one, i.e. we concatenated the prediction for all the folds and calculated the curves on them.

The results are summarized in tables 5.2 to 5.5, one table for each task, one column for each model described in section 4.2. Rows specify the metrics described in appendix A, with also macro and weighted average for precision, recall and F_1 score. Micro averages are not visualized because equivalent to accuracy. Values are expressed in percentage, indicating average and standard deviation among folds.

Table 5.2: Results for Main-site task.

		U-SVM	*B-SVM*	*B-LSTM*	*G-CRNN*	*G-LSTM*
accuracy		89.8 ± 2.0	89.8 ± 2.0	88.6 ± 2.0	90.0 ± 1.6	90.5 ± 1.6
kappa		88.5 ± 2.2	88.6 ± 2.3	87.2 ± 2.3	88.9 ± 1.8	89.3 ± 1.8
MAPs		93.0 ± 1.5	93.0 ± 1.5	92.2 ± 1.5	93.5 ± 1.2	93.8 ± 1.1
MAPc		61.6 ± 3.9	61.3 ± 4.0	55.7 ± 3.7	62.7 ± 3.5	64.1 ± 4.1
pre.	**ma.**	65.5 ± 4.8	64.7 ± 3.2	55.0 ± 2.8	61.5 ± 3.4	61.8 ± 3.7
	we.	88.7 ± 2.0	88.8 ± 2.0	87.8 ± 1.8	89.2 ± 1.6	89.5 ± 1.7
rec.	**ma.**	55.7 ± 4.1	54.7 ± 3.8	51.6 ± 3.2	56.5 ± 3.0	58.1 ± 3.5
	we.	89.8 ± 2.0	89.8 ± 2.0	88.6 ± 2.0	90.0 ± 1.6	90.5 ± 1.6
f1s.	**ma.**	58.4 ± 4.1	57.5 ± 3.6	52.1 ± 3.1	57.0 ± 2.7	58.2 ± 3.3
	we.	88.9 ± 2.0	89.0 ± 2.1	88.0 ± 2.0	89.3 ± 1.6	89.7 ± 1.7

Table 5.3: Results for Full-site task.

		U-SVM	*B-SVM*	*B-LSTM*	*G-CRNN*	*G-LSTM*
accuracy		68.4 ± 2.3	68.7 ± 2.0	67.4 ± 1.7	70.1 ± 2.1	70.9 ± 2.0
kappa		66.5 ± 2.4	66.8 ± 2.1	65.6 ± 1.7	68.4 ± 2.2	69.3 ± 2.1
MAPs		78.4 ± 1.9	78.4 ± 1.7	78.5 ± 1.3	80.6 ± 1.4	81.3 ± 1.4
MAPc		43.1 ± 2.2	43.4 ± 2.2	36.8 ± 2.3	42.9 ± 2.6	45.0 ± 2.0
pre.	**ma.**	41.4 ± 1.6	41.6 ± 1.5	33.0 ± 2.8	38.7 ± 3.1	39.8 ± 2.3
	we.	66.3 ± 1.9	67.1 ± 1.7	66.1 ± 1.3	68.8 ± 1.9	69.5 ± 1.5
rec.	**ma.**	35.7 ± 1.9	35.1 ± 2.1	32.0 ± 2.5	36.6 ± 3.0	38.0 ± 2.2
	we.	68.4 ± 2.3	68.7 ± 2.0	67.4 ± 1.7	70.1 ± 2.1	70.9 ± 2.0
f1s.	**ma.**	36.6 ± 1.5	36.4 ± 1.7	31.2 ± 2.3	35.9 ± 2.9	37.3 ± 2.1
	we.	66.2 ± 2.1	66.8 ± 1.8	66.0 ± 1.3	68.5 ± 2.0	69.5 ± 1.8

Table 5.4: Results for Type task.

		U-SVM	*B-SVM*	*B-LSTM*	*G-CRNN*	*G-LSTM*
accuracy		81.9 ± 1.9	82.9 ± 2.0	82.8 ± 1.4	84.6 ± 1.4	84.9 ± 1.5
kappa		79.5 ± 2.2	80.7 ± 2.3	80.6 ± 1.6	82.7 ± 1.6	83.0 ± 1.7
MAPs		87.8 ± 1.3	88.6 ± 1.4	88.7 ± 1.0	90.3 ± 0.9	90.6 ± 1.0
MAPc		62.4 ± 1.6	64.4 ± 1.8	55.1 ± 3.1	64.2 ± 1.9	65.9 ± 1.9
pre.	**ma.**	56.1 ± 2.4	58.3 ± 1.9	47.0 ± 3.3	56.5 ± 1.8	57.0 ± 2.6
	we.	80.3 ± 1.8	81.8 ± 1.9	82.0 ± 1.3	84.1 ± 1.3	84.3 ± 1.5
rec.	**ma.**	51.1 ± 2.6	52.2 ± 2.2	47.0 ± 2.6	56.8 ± 2.2	58.6 ± 2.0
	we.	81.9 ± 1.9	82.9 ± 2.0	82.8 ± 1.4	84.6 ± 1.4	84.9 ± 1.5
f1s.	**ma.**	51.4 ± 2.5	52.9 ± 1.9	45.0 ± 2.9	54.6 ± 1.9	55.5 ± 2.3
	we.	80.4 ± 2.0	81.7 ± 2.0	81.9 ± 1.3	83.8 ± 1.4	84.0 ± 1.5

Table 5.5: Results for Behavior task.

		U-SVM	*B-SVM*	*B-LSTM*	*G-CRNN*	*G-LSTM*
accuracy		95.9 ± 1.0	96.0 ± 1.1	94.1 ± 3.0	94.4 ± 4.2	96.5 ± 0.8
kappa		82.3 ± 4.6	82.8 ± 5.0	70.4 ± 25.5	67.6 ± 35.9	85.6 ± 3.4
MAPs		97.7 ± 0.6	97.8 ± 0.6	96.6 ± 1.8	96.8 ± 2.5	98.1 ± 0.5
MAPc		85.4 ± 5.9	85.9 ± 5.7	71.4 ± 18.4	75.5 ± 26.4	89.5 ± 4.2
pre.	**ma.**	87.0 ± 5.0	87.9 ± 4.8	69.9 ± 19.9	72.7 ± 27.1	85.5 ± 4.0
	we.	95.8 ± 1.1	95.9 ± 1.2	92.6 ± 6.4	92.1 ± 9.0	96.6 ± 0.8
rec.	**ma.**	78.6 ± 7.3	78.6 ± 7.4	67.6 ± 17.4	72.1 ± 25.4	85.9 ± 4.9
	we.	95.9 ± 1.0	96.0 ± 1.1	94.1 ± 3.0	94.4 ± 4.2	96.5 ± 0.8
f1s.	**ma.**	81.7 ± 6.3	82.0 ± 6.3	68.0 ± 18.5	72.1 ± 26.0	85.5 ± 4.2
	we.	95.8 ± 1.1	95.9 ± 1.2	93.2 ± 4.8	93.1 ± 6.7	96.5 ± 0.8

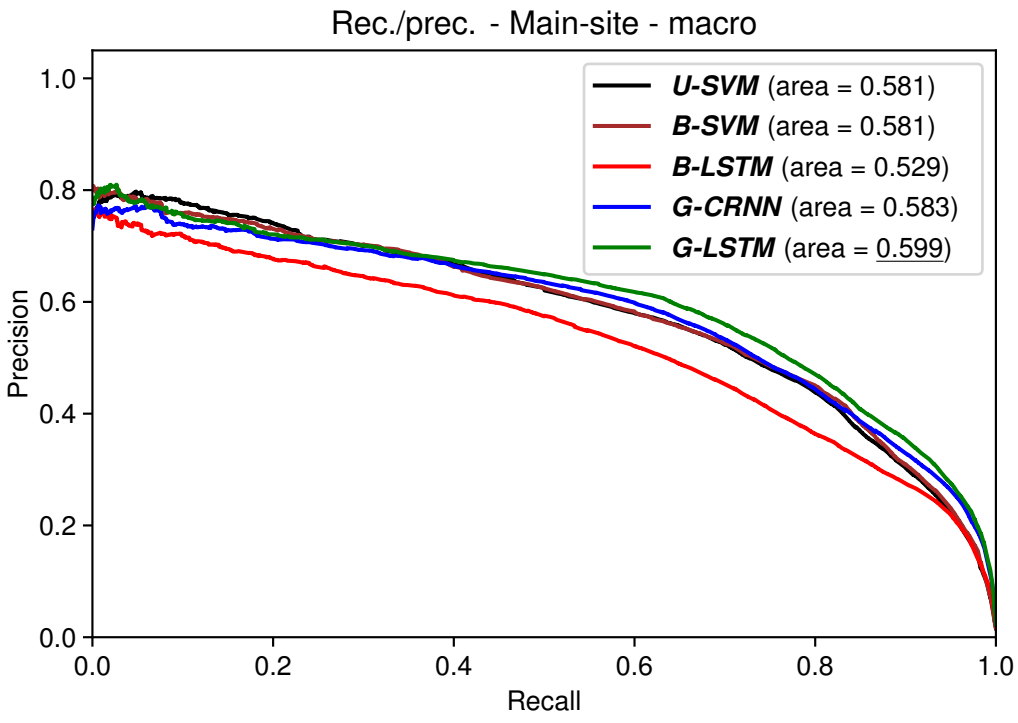


Figure 5.2: Macro-averaged recall-precision curves for Main-site task.

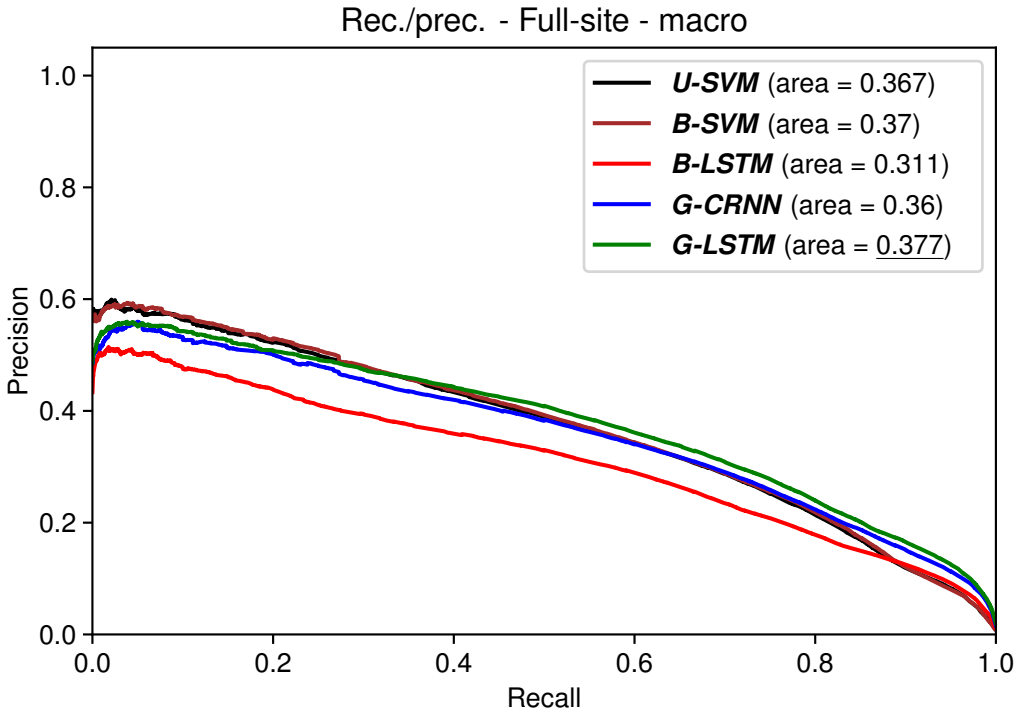


Figure 5.3: Macro-averaged recall-precision curves for Full-site task.

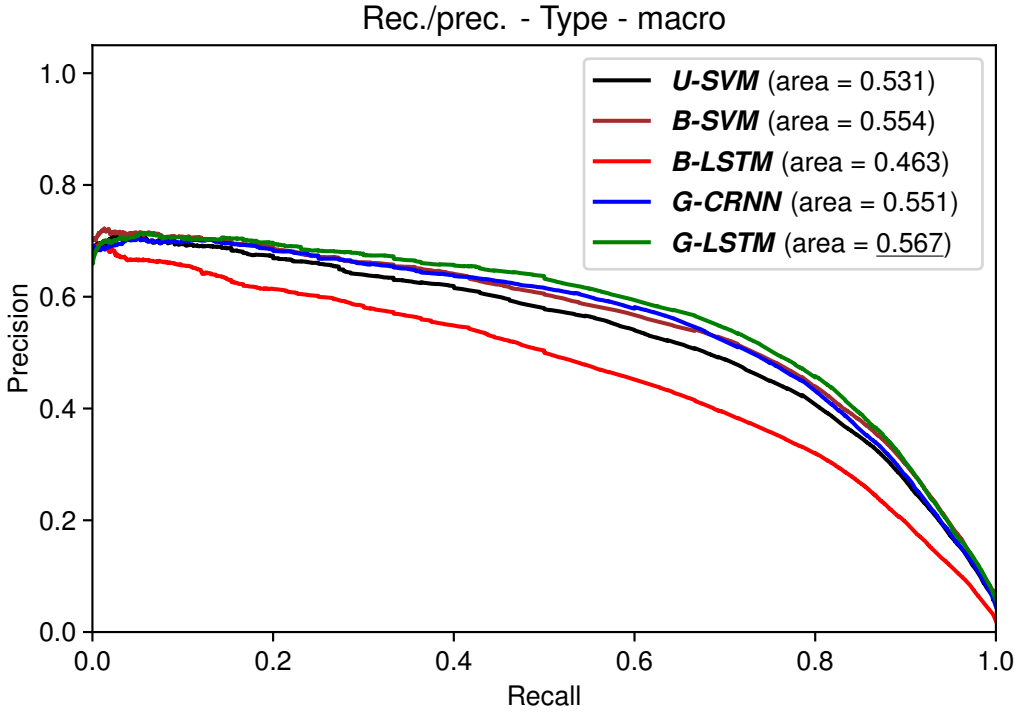


Figure 5.4: Macro-averaged recall-precision curves for Type task.

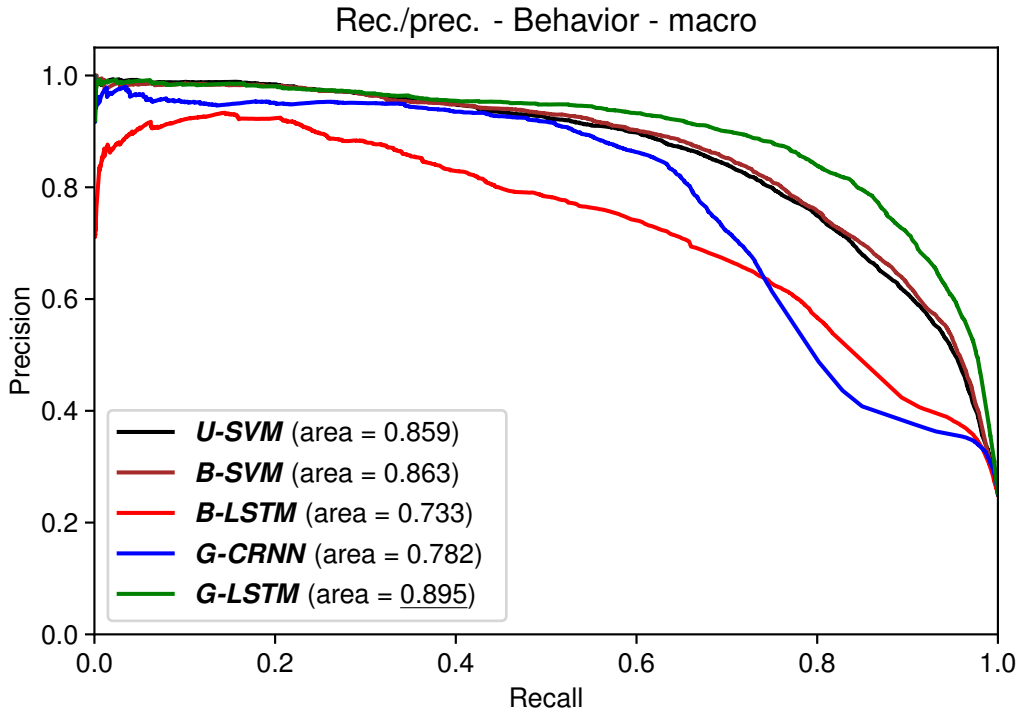


Figure 5.5: Macro-averaged recall-precision curves for Behavior task.

Macro-averaged recall-precision curves for every method and task are summarized in figs. 5.2 to 5.5. Areas under the curves are indicated in plot legends.

One of the aims of these experiments was to investigate different machine learning models for cancer cases classification based on the interpretation of pathological-reports free text. We can state that Considering SVM models, the improvement of using bigrams respect to monograms is not remarkable. The use of deep learning is not necessarily beneficial to the classification task, we already obtained good results using SVM models. We can also observe that, when we took advantage of the unlabeled data with GloVe, we achieved an improvement. ***G-LSTM*** performs always better than the other models, except regarding macro-averaged precision, where SVM models are better.

This work demonstrated that the machine learning approaches can be used to provide an automated support in cancer classification based on the information contained in free-text pathology reports.

5.2 Preliminary aggregation and interpretability

Classic RNN approaches exhibit some limits related to memory. We designed an artificial experiment in order to investigate how RNN and ***G-MAX*** address those problems. The purpose of these experiments is to compare compare plain RNN

models with attention-based models. Moreover, we also want to compare the attention with the max-pooling based models, and start to explore the potentiality of the interpretable setting. The dataset $\mathcal{D} = (\boldsymbol{X} \in [0, \dots, 9]^{n \times m}, \boldsymbol{y} \in [0, 1]^n)$ is composed of n sequences of length m of digits $x_{i,j} \in [0, \dots, 9]$. If the i-th sequence contains at least three consecutive digits $x_{i,j-1}, x_{i,j}, x_{i,j+1}$ that concatenated represent a prime number, then the sample is positive and labeled with $y_i = 1$, otherwise is negative and $y_i = 0$.

We realized a series $\mathcal{D}^{(m)} = (\boldsymbol{X}^{(m)}, \boldsymbol{y}^{(m)})$ of balanced datasets of increasing complexity. Each $\mathcal{D}^{(m)}$ has 100 000 samples with sequence lengths of

$$m \in [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000].$$

We chose this setting because we wanted to underline the memorizing capability of GRU. In fact, the network needs to keep track in his memory of all the sequence, and the larger the sequence, the higher the amount of memory needed.

We trained four models on all the datasets: a plain GRU, a ***G-MAX***, a ***G-ATT***, and a ***G-MAXi***, all with the same hidden dimension of 32.

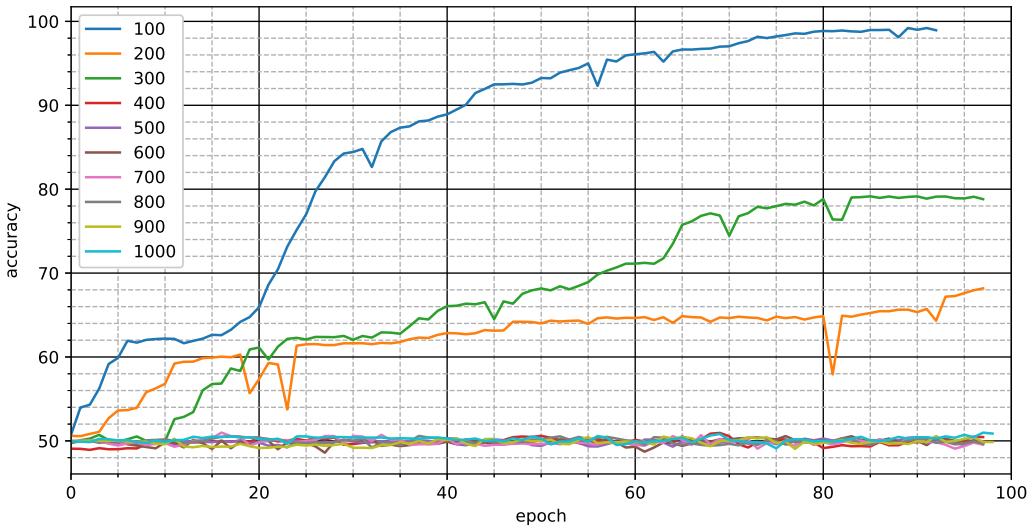


Figure 5.6: Accuracy of plain GRU on $\mathcal{D}^{(m)}$ for different dimensions of m.

In fig. 5.6 and fig. 5.7, we compare the learning curves of the two model trained on the increasing difficulty problems. We can observe that ***G-MAX*** degrades less than the baseline under the assumption of having the same dimensionality. The interpretation for this can be that the max pooling forces locality on the sequence, thus simplifying the task for the underlying RNN that at time t needs to keep track only of the neighbors of t in the sequence. Thus, ***G-MAX*** can be used on increasingly long sequences without losing potentiality.

Regarding the interpretable models, we can observe that the degradation of the interpretable ***G-MAXi*** model in fig. 5.8 is worse compared to ***G-MAX*** in fig. 5.7,

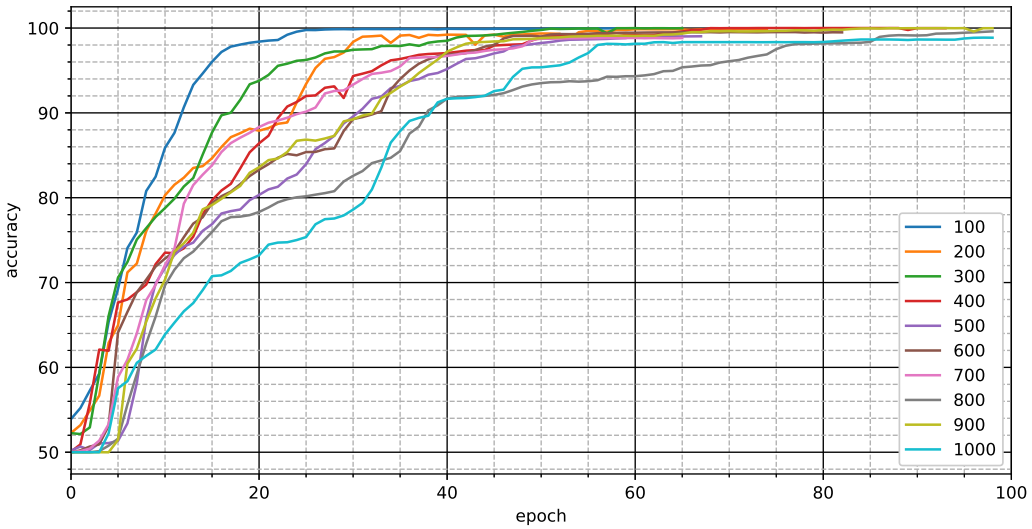


Figure 5.7: Accuracy of ***G-MAXi*** on $\mathcal{D}^{(m)}$ for different dimensions of m.

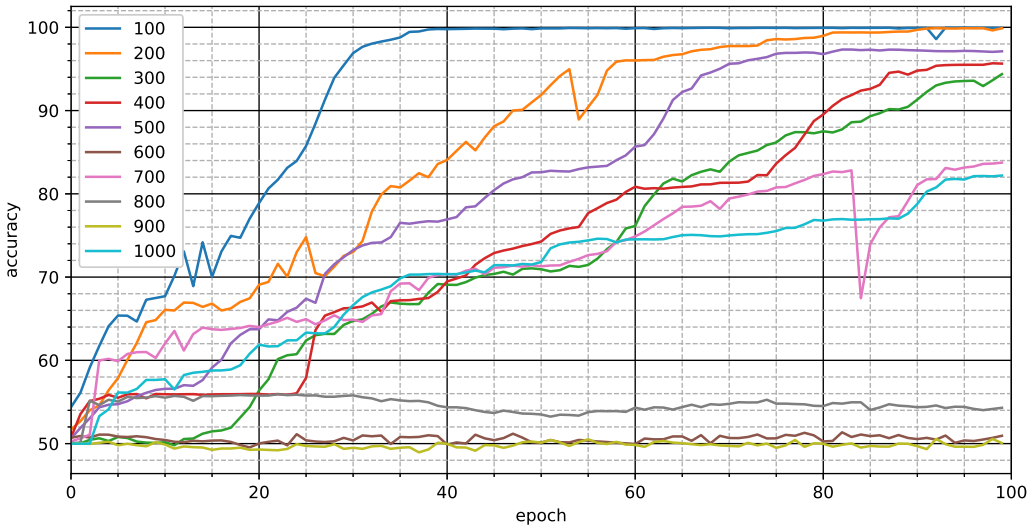


Figure 5.8: Accuracy of ***G-MAXi*** on $\mathcal{D}^{(m)}$ for different dimensions of m.

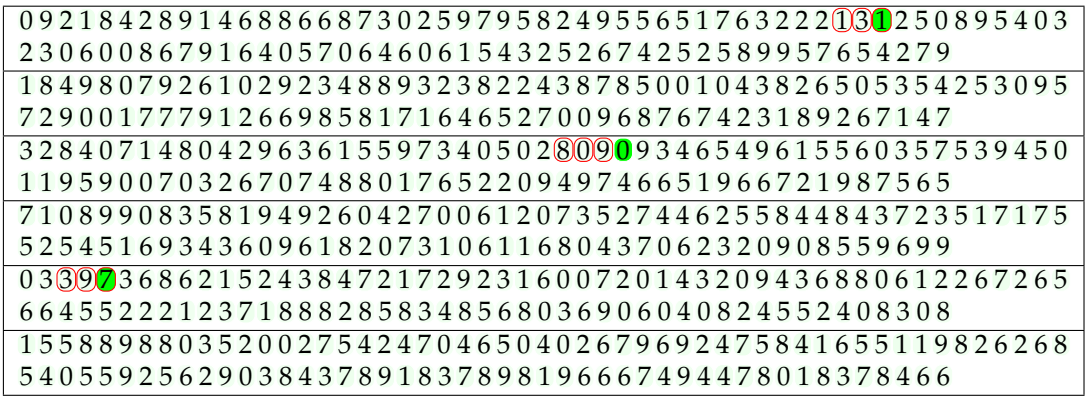


Figure 5.9: Visualization of outputs prior to the max pooling of ***G-MAXi*** for some samples. Red boxes represent the prime numbers ground truth, green highlighting represents the values of u_t in (4.4).

but is still better compared to the base GRU model in fig. 5.6. Conversely, the interpretable model can be used to gain some insights on the decision process. As visible in fig. 5.9, the model performs the classification task focusing on the part of the sequences where the prime numbers are present. The reasons why ***G-MAXi*** is not scaling as ***G-MAX*** can be found in the fact that in the interpretable setting we basically moved the aggregation from before to after the classification part. The max pooling in ***G-MAXi*** does not have the same localizing effect on the underlying part. Nevertheless, it still has some effect on simplifying the task.

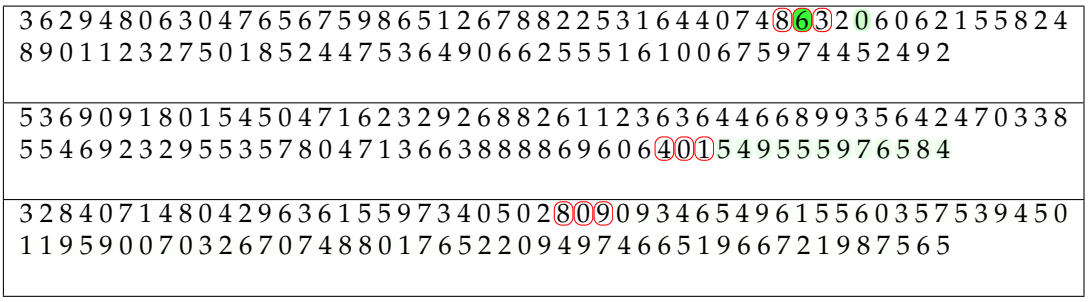


Figure 5.10: Visualization of weighted features after softmax in ***G-ATTi*** for some positive samples. Red boxes represent the prime numbers ground truth, green highlighting represents the values of $a_t(\boldsymbol{u}; \theta^a)u_t$ in (4.5).

We wanted also to compare the difference of the max vs attention aggregation. In fig. 5.10 we have empirical evidence that the attention aggregation is not sufficient to guarantee the model interpretability (at least in this setting). This can be explained with the fact that the softmax function does not avoid the learning of underlying distribute representations. On the loss calculation, the effects of a distribution with high variance before the aggregation are the same to the ones of a distribution with lower variance. Thus, the SGD does not favor focused representations in u_t.

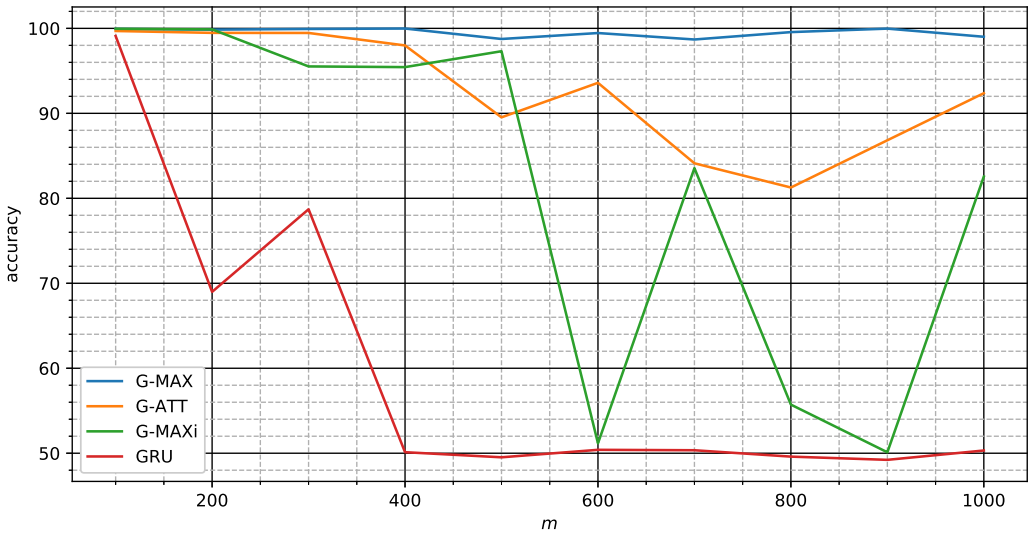


Figure 5.11: Max reached accuracy of ***G-MAX***, ***G-ATT***, ***G-MAXi***, and base GRU models on $\mathcal{D}^{(m)}$ for different dimensions of m.

In fig. 5.11, we show the maximum reached accuracy for the models, summarizing figs. 5.6 to 5.8 and also adding ***G-ATT***. From this plot we can evince that ***G-ATT*** degrades faster than ***G-MAX***. This can be due to the fact that the attention, using a softmax on all the underlying representations, still needs to rely on the entire sequence. Thus, there is not the strong focusing effect of the max aggregation.

5.3 Aggregation, interpretability, hierarchy

In these experiments we further refined the dataset. We removed duplicate reports and reports labeled with extremely rare (1048 samples that do not appear in all three of training, validation, and test sets) ICD-O3 codes. In the end we obtained a dataset suitable for supervised learning consisting of 85 170 labeled records, over 203 morphological classes, and 68 topological classes.

We defined two multi-class classification tasks: (1) main tumor site prediction (68 mutually exclusive classes) and (2) morphology prediction (203 mutually exclusive classes). Although the two tasks maybe somewhat correlated, we did not attempt multi-task approaches given the small number of tasks and large enough size of the dataset.

We decided to arrange our experiments on cancer data in a setting that simulates a predictive task. We sorted the medical records by insertion date, then we used the most recent 80% of records as test dataset, an equal amount of the remaining most recent records as validation dataset, and the rest as training dataset. The splitting of data resulted in 51 101 records for training, 17 034 for validation, and 17 035 for test

datasets.

We used the text in the 1.5 millions unlabeled records, plus the text in training datasets to train GloVe word vectors representations.

We trained the models ***G-MAX***, ***G-ATT***, ***G-MAXi***, ***G-MAXh***, and ***G-ATTh*** on both prediction tasks. We found the hyperparameters configurations doing grid search on the space explained in section 5.3.

We trained the models minimizing categorical cross entropy with Adam [28] using a starting learning rate of 0.001. The experiments were performed in *PyTorch* on machines with *GeForce RTX 2080 Ti* GPU using batches of 32 samples.

Hyperparameters

The hyperparameters of (4.1) - (4.6) and (4.13) - (4.21) control the structure of the model.

ξ^e is associated with the embedding layer E and in our case refers to GloVe hyperparameters [39]. With an intrinsic evaluation, we found that the better configuration was 60 for the vector size, 15 for the window size, and 50 iterations. ξ^f, ξ^r, $\bar{\xi}^f$, and $\bar{\xi}^r$ define the number of GRU layers ($\xi_{(l)}$) and the number of unit per each layer ($\xi_{(d)}$) respectively for F, R, $\bar{F}$, and $\bar{R}$. G is a MLP, ξ^h controls the number of layers and their size. Regarding F, R, and G, we decided to have all the stacked layer with the same size to limit the hyperparameters space. ξ^a and $\bar{\xi}^a$ control the kind of aggregating function of A and $\bar{A}$ respectively and, in case of *attention*, it controls the size of the attention layer. Finally, ξ^c controls the data-dependent output size of g.

Optimal values were obtained by grid search using the validation accuracy as the objective. In ***G-MAX*** we used the *max* aggregation function in the plain model of section 4.2.

$$
\begin{aligned}
\xi^f_{(l)} = \xi^r_{(l)} &\in [\underline{1}, 2], \\
\xi^f_{(d)} = \xi^r_{(d)} &\in [2, 4, 8, 16, 32, 64, \underline{128}, 256, 512], \\
\xi^h_{(l)} &\in [\underline{1}, 2, 4], \\
\xi^h_{(d)} &\in [2, 4, 8, 16, 32, 64, 128, 256, \underline{512}, 1024, 2048],
\end{aligned}
$$

for the *topography* site task, and:

$$
\begin{aligned}
\xi^f_{(l)} = \xi^r_{(l)} &\in [\underline{1}], \\
\xi^f_{(d)} = \xi^r_{(d)} &\in [2, 4, 8, 16, 32, 64, \underline{128}, 256, 512], \\
\xi^h_{(l)} &\in [\underline{1}, 2, 4], \\
\xi^h_{(d)} &\in [2, 4, 8, 16, 32, 64, \underline{128}, 256, 512, 1024, 2048],
\end{aligned}
$$

for the *morphology* type task.

In ***G-ATT*** we used the *attention* aggregation function in the plain model. The hyperparameters space was:

$$\begin{aligned}
\xi^f_{(l)} = \xi^r_{(l)} &\in [\underline{1}], \\
\xi^f_{(d)} = \xi^r_{(d)} &\in [64, \underline{128}, 256], \\
\xi^h_{(l)} &\in [0, \underline{1}], \\
\xi^h_{(d)} &\in [256, \underline{512}, 1024], \\
\xi^a_{(d)} &\in [128, \underline{256}, 512, 1024],
\end{aligned}$$

for the site, and:

$$\begin{aligned}
\xi^f_{(l)} = \xi^r_{(l)} &\in [\underline{1}], \\
\xi^f_{(d)} = \xi^r_{(d)} &\in [64, 128, \underline{256}], \\
\xi^h_{(l)} &\in [0, \underline{1}], \\
\xi^h_{(d)} &\in [64, \underline{128}, 256], \\
\xi^a_{(d)} &\in [128, \underline{256}, 512, 1024],
\end{aligned}$$

for the morphology.

In ***G-MAXh*** we used the *max* aggregation in the hierarchical model of section 4.2. The hyperparameters space was:

$$\begin{aligned}
\xi^f_{(l)} = \xi^r_{(l)} = \bar{\xi}^f_{(l)} = \bar{\xi}^r_{(l)} &\in [\underline{1}], \\
\xi^f_{(d)} = \xi^r_{(d)} = \bar{\xi}^f_{(d)} = \bar{\xi}^r_{(d)} &\in [32, \underline{64}, 128, 256], \\
\xi^h_{(l)} &\in [0, 1, \underline{2}, 4], \\
\xi^h_{(d)} &\in [256, 512, \underline{1024}, 2048],
\end{aligned}$$

for the topography, and:

$$\begin{aligned}
\xi^f_{(l)} = \xi^r_{(l)} = \bar{\xi}^f_{(l)} = \bar{\xi}^r_{(l)} &\in [\underline{1}], \\
\xi^f_{(d)} = \xi^r_{(d)} = \bar{\xi}^f_{(d)} = \bar{\xi}^r_{(d)} &\in [32, \underline{64}, 128, 256], \\
\xi^h_{(l)} &\in [0, \underline{1}, 2, 4], \\
\xi^h_{(d)} &\in [256, 512, \underline{1024}, 2048],
\end{aligned}$$

for the morphology.

In ***G-ATTh*** we used the *attention* aggregation in the hierarchical model. The hyperparameters space was:

$$\begin{aligned}
\xi^f_{(l)} = \xi^r_{(l)} = \bar{\xi}^f_{(l)} = \bar{\xi}^r_{(l)} &\in [\underline{1}], \\
\xi^f_{(d)} = \xi^r_{(d)} = \bar{\xi}^f_{(d)} = \bar{\xi}^r_{(d)} &\in [32, 64, \underline{128}, 256], \\
\xi^h_{(l)} &\in [0, \underline{1}, 2, 4], \\
\xi^h_{(d)} &\in [256, 512, \underline{1024}, 2048], \\
\xi^a_{(d)} = \bar{\xi}^a_{(d)} &\in [64, \underline{128}, 256, 512],
\end{aligned}$$

for the topography, and:

$$\begin{aligned}
\xi^f_{(l)} = \xi^r_{(l)} = \bar{\xi}^f_{(l)} = \bar{\xi}^r_{(l)} &\in [\underline{1}], \\
\xi^f_{(d)} = \xi^r_{(d)} = \bar{\xi}^f_{(d)} = \bar{\xi}^r_{(d)} &\in [32, \underline{64}, 128, 256], \\
\xi^h_{(l)} &\in [0, \underline{1}, 2, 4], \\
\xi^h_{(d)} &\in [256, 512, \underline{1024}, 2048], \\
\xi^a_{(d)} = \bar{\xi}^a_{(d)} &\in [64, \underline{128}, 256, 512],
\end{aligned}$$

for the morphology.

In ***G-MAXi*** we used the *max* aggregation in the plain model. Also we set the model to be interpretable. The hyperparameters space was:

$$\begin{aligned}
\xi^f_{(l)} = \xi^r_{(l)} &\in [1, \underline{2}, 4], \\
\xi^f_{(d)} = \xi^r_{(d)} &\in [2, 4, 8, 16, 32, 64, \underline{128}, 256, 512], \\
\xi^h_{(l)} &\in [\underline{1}, 2, 4], \\
\xi^h_{(d)} &\in [2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048],
\end{aligned}$$

for the topography, and:

$$\begin{aligned}
\xi^f_{(l)} = \xi^r_{(l)} &\in [1, \underline{2}, 4], \\
\xi^f_{(d)} = \xi^r_{(d)} &\in [64, 128, \underline{256}, 512], \\
\xi^h_{(l)} &\in [\underline{1}], \\
\xi^h_{(d)} &\in [],
\end{aligned}$$

for the morphology. Note that, in this setting, the size of the last layer of G must be equal to the output size of the model (and the softmax is applied directly after the

aggregation A, without any layer). Thus, $\xi^h_{(d)}$ refers only to the layers before the last one, if they exist.

Regarding ***G-GRU***, we searched in a space of $[1, 2, 4]$ number of layers of dimension in $[128, 256, 512, 1024]$. We found that the best configuration was using 2 layers of dimension 256.

Results

Table 5.6: Topography site prediction (61 classes)

	Accuracy	Top 3 Acc.	Top 5 Acc.	MacroF1
U-SVM	89.7	95.9	96.8	60.0
B-XGB	89.1	95.8	97.2	58.0
G-GRU	89.9	96.5	97.7	58.3
BERT	89.9	96.3	97.8	56.6
G-MAXi	88.0	95.4	96.2	46.1
G-MAXh	89.9	96.2	97.8	58.8
G-ATTh	89.9	96.3	97.7	58.0
G-MAX	**90.3**	**96.6**	**98.1**	**61.9**
G-ATT	90.1	96.2	97.6	60.0

Table 5.7: Morphology type prediction (134 classes)

	Accuracy	Top 3 Acc.	Top 5 Acc.	Macro F1
U-SVM	82.4	94.0	95.6	53.7
B-XGB	84.1	94.4	96.5	59.6
G-GRU	83.3	94.6	96.6	55.2
BERT	84.3	93.2	94.9	51.1
G-MAXi	73.4	91.0	93.6	31.3
G-MAXh	83.7	94.4	96.4	54.5
G-ATTh	83.7	94.4	96.2	57.5
G-MAX	84.6	**95.0**	**96.9**	59.2
G-ATT	**84.8**	94.9	**96.9**	**61.3**

In table 5.6 and table 5.7 we summarize the results of different models on test data in terms of accuracy, top-3 and top-5 accuracy (counting as correctly classified a document whose true class appears within the first three or five top predictions), and macro-averaged F1 score. In table 5.8 we investigate the $F1$ score on different subsets of classes. We consider a class *easy* if it has more than 1000 examples in the test set, *average* if it has between 100 and 1000 examples, and *hard* if it has less than 100 examples.

Table 5.8: Macro F1 measure by groups of class frequency

	Topography			Morphology		
	easy (4 cls)	avg. (18 cls)	hard (39 cls)	easy (5 cls)	avg. (18 cls)	hard (111 cls)
U-SVM	95.7	**86.9**	50.9	90.5	68.6	48.4
B-XGB	95.6	86.4	48.2	92.0	72.4	54.8
G-GRU	**96.1**	72.2	48.0	91.4	71.6	49.7
BERT	95.7	73.2	44.9	**92.9**	**74.4**	43.9
G-MAXi	95.0	66.6	31.4	87.1	41.9	25.1
G-MAXh	95.8	72.4	48.8	92.7	71.8	48.8
G-ATTh	96.0	73.1	47.1	91.9	72.3	52.6
G-MAX	96.0	73.3	**53.1**	92.7	72.3	53.8
G-ATT	96.0	73.1	50.3	92.8	72.3	**56.7**

In the case of topography, when focusing on the performance on classes with many examples, all models tend to perform similarly, with even the interpretable model attaining high F1 scores. The advantage of recurrent networks over bag-of-word representations is more pronounced when focusing on rare classes. One possible explanation is that the representation learned by recurrent networks is shared across all classes, leveraging the advantage of multi-task learning [6] in this case. We also note that in no case hierarchical attention models outperform flat attention models and max-pooling performs the best on rare classes. In the case of morphology, differences among different models are more pronounced, with BERT being very effective for densely populated classes (but not for rare classes). Again hierarchical attention does not outperform flat attention. This result differs from the ones reported in [21] but datasets are very different in size and number of classes, and of course differences in the writing style of pathologist in our dataset could be significant. In particular our documents contains on average few sentences (see fig. 4.1), and this can be the main reason of the poor performances of hierarchical models.

The interpretable model is not as powerful as ***G-MAX***, and in this task not even as the baseline, but it can be used as a classification support and to gain insight in the classification process. In table 5.9, we show three different samples where we underline with different colors — only for the indicated relevant codes — the values of $\boldsymbol{h}_{i,j}$ in (4.4). We consider a code relevant if the corresponding value in the 68-dimension vector $\boldsymbol{h}_{i,j}$ is greater than 0.1. In the first sample, that was correctly classified, all the terms related to prostate gland cancer are strongly underlined. Apart from *prostatico* and *prostata* that are Italian terms for *prostatic* and *prostate*, the main underlined terms are *PSA* (Prostate-Specific Antigene) and *Gleason* score that are two common exams in prostate cancer cases [4].

The second sample is a more difficult document, it is roughly translated in en-

Table 5.9: Visualization of interpretable outputs.

y_i	Relevant $h_{i,j}$	$x_{i,j}$, relevant $h_{i,j}$
61	61 (PROSTATE GLAND)	DISOMOGENICITA ' DIFFUSE . PSA NON PERVENUTO . ADENOCARCINOMA PROSTATICO A GRADO DI DIFFERENZIAZIONE MEDIO - BASSO (GLEASON 3 + 4) NEI PRELIEVI DI CUI AI NN . 2 E 3 . AGOBIOPSIA DELLA PROSTATA : 1) 1 PRELIEVO LL DX . 2) 2 PRELIEVI ML DX . 3) 2 PRELIEVI M DX . 4) 1 PRELIEVO M SX . 5) 2 PRELIEVI ML SX . 6) 1 PRELIEVO LL SX . 7) 1 PRELIEVO TRANSIZIONALE SX . 8) 1 PRELIEVO TRANSIZIONALE DX .
20	18 (COLON) 20 (RECTUM) 21 (ANUS AND ANAL CANAL)	ISOLATI FRAMMENTI RIFERIBILI AD ADENOMA TUBULARE INTESTINALE DI ALTO GRADO . FRAMMENTI (NR . 2) DI POLIPO PEDUNCOLATO A 20 CM DALL ' ORIFIZIO ANALE . (ESEGUITA COLORAZIONE EMATOSSILINA - EOSINA) .
34	34 (BRONCHUS AND LUNG) 56 (OVARY) 67 (BLADDER) 80 (UNKNOWN PRIMARY SITE)	VERSAMENTO PLEURICO SX DI N . D . D . E ADDENSAMENTI POLMONARI DI N . D . D . , NODULI PARETE ADDOMINALE . INFILTRAZIONE CANCERIGNA DEGLI STROMI CONNETTIVO - ADIPOSI . IMMUNOISTOCHIMICA : CK7 + , CK20 - , TTF - 1 - , PROTEINA S - 100 - . LESIONE DI CM 2 , 0 X 1 , 3 X 0 , 7 . 1 - 2) SEZIONI SERIATE .

glish as:

ISOLATED FRAGMENTS ATTRIBUTABLES
TO HIGH DEGREE
INTESTINAL TUBULAR ADENOMA.
FRAGMENTS (NR. 2) OF PEDUNCULATED
POLYPUS AT 20 CM FROM
THE ANAL ORIFICE. (PERFORMED
HEMATOXYLIN-EOSIN COLORING).

For this sample, the model propose the three classification codes *18*, *20*, and *21* (with value 1 for both *18* and *20*, and little less for *21*). It suggests that the terms *intestinal tubular adenoma* and *pedunculated polypus* are related to colon, *polypus* can be related also to rectum, and *anal orifice* is related to rectum and anus. Note that this record was labeled from the RTT with the code for rectum, while the medical report explicitly mentions that the fragments have been extracted at 20 cm from the anal orifice (the human rectum is long approximately 12 cm and the anal canal 3-5 cm [22]). This record is an example of the complexity of the dataset. The mislabeling is not necessarily a human classification error, RTT have access to more information respect to the one contained in our dataset. The fact that this example refers in particular to an operation to the colon does not exclude that it was related to a rectum tumor.

The third sample is an even more complex record, in english it is translated as:

```
LEFT PLEURAL EFFUSION OF
UNKNOWN ORIGIN AND LUNG
THICKENING OF UNKNOWN ORIGIN,
NODULES OF THE ABDOMINAL WALL.
CANCEROUS INFILTRATION OF THE
CONNECTIVE-ADIPOSE STROMA.
IMMUNOHISTOCHEMICAL: CK7 +,
CK20 -, TTF-1 -, PROTEIN
S-100 -.  2 CM LESION,
0 X 1,3 X 0,7.  1-2)
SERIAL SECTIONS.
```

The model classifies the record mainly with codes *34* and *80*, and less with *56* and *67*. It underlines with the lung code the terms *plurial effusion* and *lung thickening*, but interestingly it also underlines the immunohistochemical results. The positive *CK7*, negative *CK20* pattern represents a common diagnosis of lung origin for metastatic adenocarcinoma [29]. Also, immunohistochemistry is a common approach in the diagnosis of tumors of uncertain origin [17]. This can be the reason for the underlying with code *80* of the immunoistochemical part. It is interesting to note also that *pleuric* is suggested to be related to ovary cancer, in fact the pleural cavity constitutes the most frequent site for extra abdominal metastasis in ovarian carcinoma [40].

These experiments with interpretable models serve the purpose of illustrating that these models can be useful even if they perform poorly in the classification. In a practical context, it is possible to combine the more powerful ***G-MAX*** with the interpretable ***G-MAXi***. A software that helps humans in the classification process can use ***G-MAX*** to suggest the most-probable class and ***G-MAXi*** to highlight the most relevant terms.

To quantify the effectiveness of the interpretability, we designed an experiment where a dataset is created taking for each document the first k words selected by ordering the results of the aggregator u_t in case of ***G-MAXi***, and $a_t(\boldsymbol{u}; \theta^a)u_t$ in case of ***G-ATTi***. In fig. 5.12, we plot the accuracy obtained training a plain GRU model on the cleaned datasets, for increasing values of k. Surprisingly, even with few words the document is classified practically with the same results as ***G-GRU***. This can mean both that the interpretation is working well distilling the document with its most relevant terms, and that the information contained in our medical texts is concentrated in few terms. The latter can also be the reason why in tables 5.2 and 5.4 we do not observe a big improvement in using deep learning methods respect to SVMs.

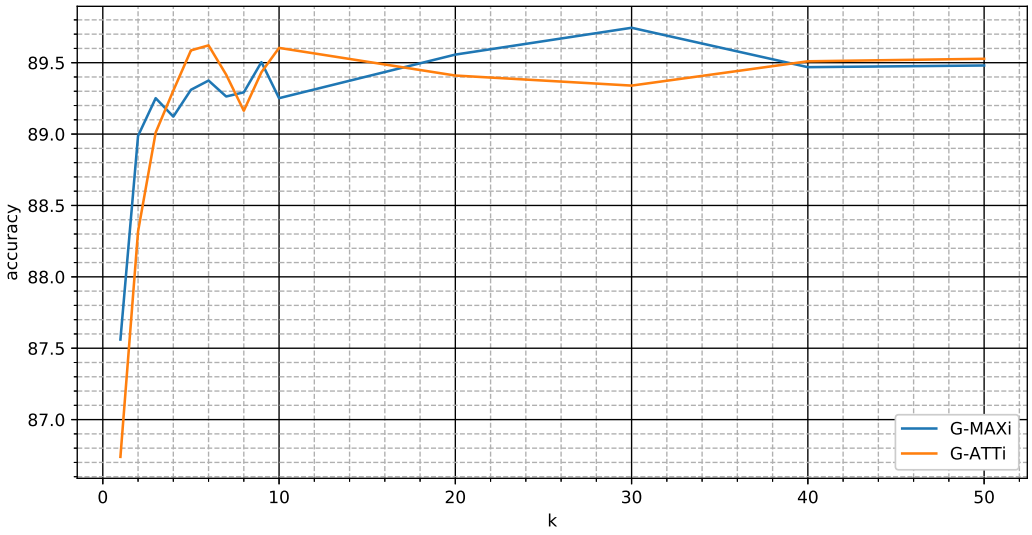


Figure 5.12: Training of a plain GRU model on a dataset created using ***G-MAXi*** and ***G-ATTi*** to keep the first k words.

Chapter 6

Conclusions

Since the cancer registration process is partially based on manual revision, including also the interpretation of the free text in pathological reports, significant delays in data production and publication may occur. This weakens data relevance for the purpose of assessing compliance with updated regional recommended integrated case pathways, as well as for public health purposes. Improving automated methods to generate a list of putative incident cases and to automatically estimate process indicators is thus an opportunity to perform an up-to-date evaluation of cancer-care quality. In particular, machine learning techniques like the ones presented in this paper could overcome the delay in cancer case definition by the cancer registry and allow a powerful tool for timely indicators computation. The implementation of this procedure could guarantee an automated and validated instrument to monitor and evaluate diagnostic and therapeutic pathways.

We analyzed the available data and created different models in order to implement an automated classification system. We obtained very encouraging results in classifying cancer cases based on the interpretation of free text in the data-flow of pathology reports. This suggests that machine learning methods can be usefully leveraged in this context. Moreover, we demonstrated that unlabeled data can be effectively used to construct useful word vectors and improve classification accuracy.

Our models also have the added value that they can be utilized to retrieve records adjusting the precision-recall trade-off.

The use of administrative data sources that are up to date combined with powerful machine learning techniques to automate text classification is in the interest of the development of a standardized surveillance system at Regional and National level. Stakeholders and decision makers need timely and updated indicators to evaluate and plan healthcare activities. The availability of timely indicators, routinely and automatically produced, is technically possible. The main novelty of this work is to show the power of machine learning techniques applied to the classification of

free text pathological records. This was not yet been systematically implemented in other Italian cancer registries. This provides a useful monitor tool for cancer patients pathways, allowing to describe population's general health state and to establish public health goals.

The results of the interpretable models can be used to assist the human classification process on simple records. It can be used as a form of text compression, highlighting the most important terms. On more complex records it can be used to leverage the knowledge of the model to gain insight on the decision process. To overcome the limitations of the interpretable model respect to the general model, in terms of classification metrics, is it possible to combine the two variants. The general model can be used to give a more authoritative classification on the samples while at the same time, the interpretable model can highlight the same samples.

We compared novel deep learning techniques and more classical models to pathology records. In this specific context we did not obtain significant improvements using novel deep learning approaches respect to classic machine learning methods. Also, the attention methods usually employed in text classification tasks do not have better results respect to a more simple max pooling hard attention. Furthermore, hierarchical models do not work better than plain models. Unsupervised methods, in particular Word Vectors, can be used successfully in the domain of pathological text records. At the best of our knowledge, this is the first large scale study of deep learning methods applied to pathology records. Other studies where performed on smaller datasets with records labeled with less classes.

Regarding the questions in section 1.4, Q1 is answered by the fact that we implemented several different models to a large scale dataset of cancer pathology reports. Q2 is answered by the fact that we used attention models and BERT in our experiments. To answer to Q3, from our experiments we have evidence that by using deep learning methods we do not have a breakthrough compared to classic ML approaches in this specific domain. Regarding Q4, we observe that hierarchical models are not beneficial in this context. Moreover, we achieve a little improvement by using attention models, but in this context a simple max aggregation is equally powerful to the commonly used attention. About Q5 we observe a successful improvement when we leverage the unlabeled data, thus we can conclude that unsupervised techniques can be successfully used in this context. Finally, in relation to Q6 we studied the potentialities of interpretable models in the pathology records context.

Appendix A

Metrics

We used different metrics in order to evaluate the models. All the metrics are defined between 0 and 1 or between -1 and 1, the higher the value, the better the assessment.

A.1 Accuracy

is defined as the ratio between the correct-classified documents and all documents. If $\boldsymbol{y}$ is the ground truth and $\hat{\boldsymbol{y}}$ is the predicted classification vectors for n samples, then the accuracy is defined as:

$$accuracy\left(\boldsymbol{y}, \hat{\boldsymbol{y}}\right) \equiv \frac{1}{n} \sum_{i=1}^{n} 1\left(\hat{y}_i = y_i\right),$$

where

$$1(a = b) \equiv \begin{cases} 1 & \text{if } a = b, \\ 0 & \text{otherwise.} \end{cases}$$

In an unbalanced dataset, like the one of this work, it is a biased score - a model that predicts well only the most frequent classes, and ignores the rest, achieves a good accuracy. To resolve this we also considered other metrics.

A.2 Cohen's kappa

score is usually used to asses the agreement of two annotators [11]. It measures the difference between the observed agreement and the agreement that can happen by choosing randomly the class. It can then be used to mitigate the bias caused by the unbalanced dataset. Cohen's kappa is defined as:

$$\kappa(\boldsymbol{y}, \hat{\boldsymbol{y}}) \equiv \frac{p_o(\boldsymbol{y}, \hat{\boldsymbol{y}}) - p_e(\boldsymbol{y}, \hat{\boldsymbol{y}})}{1 - p_e(\boldsymbol{y}, \hat{\boldsymbol{y}})} = 1 - \frac{1 - p_o(\boldsymbol{y}, \hat{\boldsymbol{y}})}{1 - p_e(\boldsymbol{y}, \hat{\boldsymbol{y}})},$$

where p_o is the observed agreement that is equal to the accuracy and p_e represents the probability of agreement by chance. For n samples and k classes it is defined by:

$$p_e(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{n^2} \sum_{i=1}^{k} \mu_i(\boldsymbol{y}) \cdot \nu_i(\hat{\boldsymbol{y}}),$$

where μ_i and ν_i are the number of samples classified as i for the first and second classifier. They are defined as:

$$\mu_i(\boldsymbol{y}) = \sum_{j=1}^{n} 1(y_j = i)$$
$$\nu_i(\hat{\boldsymbol{y}}) = \sum_{j=1}^{n} 1(\hat{y}_j = i)$$

A.3 Mean Average Precision (MAP)

is a measure used in information retrieval [34]. It expresses how well the true classification can be retrieved in the first results of the classifier. We define two variants of MAP, one (MAPc) to state how well all records for a specific class are retrieved, the other (MAPs) to asses how well the correct class is retrieved for a specific sample. The first is defined for n samples and k classes, with $\boldsymbol{Y} = \boldsymbol{y}_1, \ldots, \boldsymbol{y}_k$ being the ground truth and $\hat{\boldsymbol{Y}} = \hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_k$ being the prediction. The formula is:

$$MAPc(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) \equiv \frac{1}{k} \sum_{c=1}^{k} AveP(\boldsymbol{y}_c, \hat{\boldsymbol{y}}_c),$$

where $AveP$ is the average precision for class c:

$$AveP(\boldsymbol{y}, \hat{\boldsymbol{y}}) \equiv \frac{1}{\sum_{i=1}^{k} 1(y_i)} \sum_{j=1}^{k} P_j(\boldsymbol{y}, \hat{\boldsymbol{y}}) \cdot 1(y_{\sigma_{\hat{\boldsymbol{y}}}(j)}),$$

where $1(y)$ is an indicator function that is 1 for the elements classified positively, 0 otherwise. $\sigma_{\hat{\boldsymbol{y}}}(j)$ is a function that returns the index in $\hat{\boldsymbol{y}}$ of the j-th element in the ordered version of $\hat{\boldsymbol{y}}$. P_j is defined as:

$$P_j(\boldsymbol{y}, \hat{\boldsymbol{y}}) \equiv \frac{1}{j} \sum_{c=1}^{j} 1(y_{\sigma_{\hat{\boldsymbol{y}}}(c)}).$$

MAPs is defined like MAPc, but on the transposed classification matrices

$$MAPs(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) \equiv MAPc(\boldsymbol{Y}^T, \hat{\boldsymbol{Y}}^T).$$

A.4 Precision

is defined for n samples and binary classifications as:

$$P(\boldsymbol{y}, \hat{\boldsymbol{y}}) \equiv \frac{\sum_{s=1}^{n} 1(\hat{y}_s \text{ and } y_s)}{\sum_{s=1}^{n} 1(\hat{y}_s)}. \tag{A.1}$$

It expresses the ratio of correct positive predictions over all the positive predictions.

A.5 Recall

is defined as:

$$R(\boldsymbol{y}, \hat{\boldsymbol{y}}) \equiv \frac{\sum_{s=1}^{n} 1(\hat{y}_s \text{ and } y_s)}{\sum_{s=1}^{n} 1(y_s)}, \tag{A.2}$$

and it is the ratio of correct predicted positive over all the positive classes.

A.6 F1-score

is the harmonic mean of precision and recall, combining the two measures:

$$F_1(\boldsymbol{y}, \hat{\boldsymbol{y}}) \equiv 2\frac{P(\boldsymbol{y}, \hat{\boldsymbol{y}}) \cdot R(\boldsymbol{y}, \hat{\boldsymbol{y}})}{P(\boldsymbol{y}, \hat{\boldsymbol{y}}) + R(\boldsymbol{y}, \hat{\boldsymbol{y}})}.$$

Precision, recall, and thus F_1 score are defined only for binary classifiers. In order to use these metrics in a multi-class classification problem it is possible to average the measures for the different classes. We considered different methodologies of averaging:

micro averaging is performed flattening the array of truth and prediction of the different classes and then applying the scoring formula;

macro average is performed calculating the metrics on the single classes and then averaging them:

$$\frac{1}{k} \sum_{c=1}^{k} S(\boldsymbol{y}_c, \hat{\boldsymbol{y}}_c);$$

weighted average uses the normalized number of samples for each class in order to give a weight to them:

$$\frac{1}{\sum_{i=1}^{k} |\hat{\boldsymbol{y}}_i|} \sum_{c=1}^{k} |\hat{\boldsymbol{y}}_c| \cdot S(\boldsymbol{y}_c, \hat{\boldsymbol{y}}_c).$$

Micro average considers all the samples equally, regardless of the representativeness of classes in the dataset. Macro average considers the unbalancement and it is more sensible to few represented classes. Precision, recall, and F_1 score are equal to the accuracy when micro averaged in a multiclass environment.

A.7 ROC curve

is a graph of *true positive rate* versus *false positive rate* with the change of the classifier threshold. The true positive rate is equal to the recall defined in eq. (A.2). Conversely the false positive rate is defined as:

$$FPR(\boldsymbol{y}, \hat{\boldsymbol{y}}) \equiv \frac{\sum_{s=1}^{n} 1(\hat{y}_s \text{ and not } y_s)}{\sum_{s=1}^{n} 1(\text{not } y_s)}. \tag{A.3}$$

ROC curves start from $(0,0)$ and end in $(1,1)$. The area under the curve can be used as a metric for the classifier, a perfect classifier has an area of 1.

The curves can be calculated only for binary classifiers. Like for the precision, recall and F_1 score, it is possible to generalise them to multiclass problems averaging micro or macro.

A.8 Precision-recall curve

is a graph of precision (eq. (A.1)) versus recall (eq. (A.2)) with the change of the classifier threshold. The curves start from $(0,1)$ and end in $(1,0)$. A perfect classifier has an area under the curve of 1.

Also, precision-recall curves can be calculated only for binary classifiers. In order to generalise them to multiclass problem, it is necessary to micro or macro average.

A.9 Use cases

The different metrics are useful to assess models in different situations. In case you need to retrieve records of a specific cancer case from the register, MAPc assesses how well the task is executed. Conversely in case of an operator-assistance software, MAPs assess how the correct classification for a specific histological record is retrieved on top results.

Cohen's kappa measures the agreement of automatic and human annotators. Thus, it is an indirect measure of the classification correctness.

Precision and recall are two measures in trade off between them. The former is more significant when you need a list of correctly-classified records, at cost of not retrieving all of them. The latter is more meaningful when you need to retrieve the greatest number of cases at cost of retrieving also some false positives. A peculiarity of the automatic annotator is that you can change the threshold to support higher precision or recall depending on the specific retrieving task. Recall-precision curves can be used to assess the correct threshold.

Appendix B

Publications

Journal papers

1. **Stefano Martina**, Leonardo Ventura, Paolo Frasconi, "Classification of cancer pathology reports: a large-scale comparative study", *Journal of Biomedical and Health Informatics*, in review. **Candidate's contributions**: prepared dataset, designed methods and experiments, designed algorithms

Peer reviewed conference papers

Workshop papers

Papers under review

Other

(e.g. ArXiv preprints not yet submitted)

Bibliography

[1] B. Stewart and C.P. Wild, editors (2014). *World Cancer Report 2014*. International Agency for Research on Cancer, WHO.

[2] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473 [cs, stat]*. 08741 arXiv: 1409.0473.

[3] Bottou, L. (2012). Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer.

[4] Brimo, F., Montironi, R., Egevad, L., Erbersdobler, A., Lin, D. W., Nelson, J. B., Rubin, M. A., van der Kwast, T., Amin, M., and Epstein, J. I. (2013). Contemporary grading for prostate cancer: Implications for patient care. *European Urology*, 63(5):892 – 901.

[5] Brown, M., Lipscomb, J., and Snyder, C. (2001). The burden of illness of cancer: economic cost and quality of life. *Annual Review of Public Health*, 22:91–113.

[6] Caruana, R. (1997). Multitask learning. *Machine learning*, 28(1):41–75.

[7] Chatterjee, A., Narahari, K. N., Joshi, M., and Agrawal, P. (2019). Semeval-2019 task 3: Emocontext contextual emotion detection in text. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 39–48.

[8] Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM.

[9] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[10] Codish, S. and Shiffman, R. N. (2005). A model of ambiguity and vagueness in clinical practice guideline recommendations. In *AMIA Annual Symposium Proceedings*, volume 2005, page 146. American Medical Informatics Association.

[11] Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

[12] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

[13] Crocetti, E., Sacchettini, C., Caldarella, A., and Paci, E. (2004). Automatic coding of pathologic cancer variables by the search of strings of text in the pathology reports. The experience of the Tuscany Cancer Registry. *Epidemiologia e prevenzione*, 29(1):57–60.

[14] Delen, D., Walker, G., and Kadam, A. (2005). Predicting breast cancer survivability: a comparison of three data mining methods. *Artificial Intelligence in Medicine*, 34(2):113–127.

[15] DeSantis, C. E., Lin, C. C., et al. (2014). Cancer treatment and survivorship statistics, 2014. *CA: A Cancer Journal for Clinicians*, 64(4):252–271.

[16] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[17] Duraiyan, J., Govindarajan, R., Kaliyappan, K., and Palanisamy, M. (2012). Applications of immunohistochemistry. *Journal of pharmacy & bioallied sciences*, 4(Suppl 2):S307.

[18] Fellbaum, C. (1998). *WordNet – An Electronic Lexical Database*. MIT Press.

[19] Ferretti, S., Giacomin, A., et al. (2008). *Cancer Registration Handbook*. AIRTUM.

[20] Fritz, A., Percy, C., Jack, A., Shanmugaratnam, K., Sobin, L., Parkin, D. M., and Whelan, S., editors (2000). *International classification of diseases for oncology*. World Health Organization, Geneva, 3 edition.

[21] Gao, S., Young, M. T., Qiu, J. X., Yoon, H.-J., Christian, J. B., Fearn, P. A., Tourassi, G. D., and Ramanthan, A. (2018). Hierarchical attention networks for information extraction from cancer pathology reports. *Journal of the American Medical Informatics Association*, 25(3):321–330.

[22] Greene, F., Compton, C., on Cancer, A. J. C., Fritz, A., Shah, J., and Winchester, D. (2006). *Ajcc Cancer Staging Atlas*. Springer New York.

[23] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

[24] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[25] Hu, D. (2019). An introductory survey on attention mechanisms in nlp problems. In *Proceedings of SAI Intelligent Systems Conference*, pages 432–448. Springer.

[26] Jouhet, V., Defossez, G., Burgun, A., Le Beux, P., Levillain, P., Ingrand, P., and Claveau, V. (2011). Automated Classification of Free-text Pathology Reports for Registration of Incident Cases of Cancer:. *Methods of Information in Medicine*, 51(3):242–251.

[27] Kavuluru, R., Hands, I., Durbin, E. B., and Witt, L. (2013). Automatic extraction of ICD-O-3 primary sites from cancer pathology reports. In *Clinical Research Informatics AMIA symposium (forthcoming)*.

[28] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[29] Kummar, S., Fogarasi, M., Canova, A., Mota, A., and Ciesielski, T. (2002). Cytokeratin 7 and 20 staining for the diagnosis of lung and colorectal adenocarcinoma. *British journal of cancer*, 86(12):1884.

[30] LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

[31] LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. (2006). A tutorial on energy-based learning. *Predicting structured data*, 1(0).

[32] Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., and Kang, J. (2019). Biobert: pre-trained biomedical language representation model for biomedical text mining. *arXiv preprint arXiv:1901.08746*.

[33] Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

[34] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

[35] Martinez, D. and Li, Y. (2011). Information extraction from pathology reports in a hospital setting. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1877–1882. ACM.

[36] Mikolov, T., Yih, W.-t., and Zweig, G. (2013). Linguistic Regularities in Continuous Space Word Representations. In *Hlt-naacl*, volume 13, pages 746–751.

[37] Miller, S., Guinness, J., and Zamanian, A. (2004). Name tagging with word clusters and discriminative training. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 337–342.

[38] Mujtaba, G., Shuib, L., et al. (2019). Clinical text classification research trends: Systematic literature review and open issues. *Expert Systems with Applications*, 116:494–520.

[39] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global Vectors for Word Representation. In *EMNLP*, volume 14, pages 1532–1543. 00365.

[40] PORCEL, J. M., DIAZ, J. P., and CHI, D. S. (2012). Clinical implications of pleural effusions in ovarian cancer. *Respirology*, 17(7):1060–1067.

[41] Qiu, J. X., Yoon, H.-J., Fearn, P. A., and Tourassi, G. D. (2018). Deep Learning for Automated Extraction of Primary Sites From Cancer Pathology Reports. *IEEE Journal of Biomedical and Health Informatics*, 22(1):244–251.

[42] Siegel, R. L., Miller, K. D., and Jemal, A. (2016). Cancer statistics, 2016: Cancer Statistics, 2016. *CA: A Cancer Journal for Clinicians*, 66(1):7–30.

[43] Sukhbaatar, S., Weston, J., Fergus, R., et al. (2015). End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

[44] Sullivan, R., Peppercorn, J., et al. (2011). Delivering affordable cancer care in high-income countries. *The Lancet Oncology*, 12(10):933–980.

[45] Tibo, A., Frasconi, P., and Jaeger, M. (2017). A network architecture for multi-multi-instance learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 737–752. Springer.

[46] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

[47] Tourassi, G. (2017). Deep learning enabled national cancer surveillance. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 3982–3983. 00001.

[48] Tshitoyan, V., Dagdelen, J., Weston, L., Dunn, A., Rong, Z., Kononova, O., Persson, K. A., Ceder, G., and Jain, A. (2019). Unsupervised word embeddings capture latent knowledge from materials science literature. *Nature*, 571(7763):95.

[49] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

[50] Wager, S., Wang, S., and Liang, P. S. (2013). Dropout training as adaptive regularization. In *Advances in neural information processing systems*, pages 351–359.

[51] Wang, Z., Zhang, Y., Yu, M., Zhang, W., Pan, L., Song, L., Xu, K., and El-Kurdi, Y. (2019). Multi-granular text encoding for self-explaining categorization. *arXiv preprint arXiv:1907.08532*.

[52] Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. (2016). Hierarchical Attention Networks for Document Classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics. 01296.

[53] Yim, W.-w., Yetisgen, M., Harris, W. P., and Kwan, S. W. (2016). Natural Language Processing in Oncology: A Review. *JAMA Oncology*, 2(6):797.

[54] Yin, W., Kann, K., Yu, M., and Schütze, H. (2017). Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*.

[55] Zhao, B. (2019). Clinical Data Extraction and Normalization of Cyrillic Electronic Health Records Via Deep-Learning Natural Language Processing. *JCO Clinical Cancer Informatics*, (3):1–9.

[56] Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320.