POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Graph Neural Network pipeline for unsupervised clinical document analysis

Tesi di Laurea Magistrale in

Mathematical Engineering - Ingegneria Matematica

Author: **Gabriele Moro**

Student ID: 996398

Advisor: Prof. Francesca Ieva

Co-advisors: Dott. Vittorio Torri

Academic Year: 2023-2024

Abstract

Most clinical documents are present in the form of unstructured data i.e. textual data. However, in order to apply automatic learning methods of the content, it is necessary to provide a mathematical representation of the text; this is one of the tasks of Natural Language Processing. In this thesis work, it will be analyzed the E3C dataset, which contains 10473 clinical cases, with the goal of producing clusters of documents, useful to the medical staff. The dataset does not provide labels for the documents, therefore, an unsupervised approach was required. Inspired by the recent success of the Graph Neural Networks (GNNs) in the similar field of supervised text classification, we have produced a fully unsupervised modification of the InfoGraph model to be able to create vector representations of E3C documents and through classical clustering methods group them according to their similarities. Before being subjected to the model, the documents have been represented as a homogeneous graph, in which each node represents a word in the text and the links between nodes represent the connection between words that appear adjacent in the text. To evaluate the results, in the absence of labels, we relied on the UMLS dictionary, on the preexisting method Doc2Vec, and on labels created automatically by ChatGPT. This is one of the first totally unsupervised graph neural network work, and to our knowledge, the first to deal with textual documents. The results show how the vectorial representations of the texts can derive information regarding the origin of the documents but only minimally about the medical content. Several modifications can be implemented to improve the performance of the model, confirming that the field of research under analysis may offer many opportunities for future development.

Keywords: Graph Neural Networks, Unsupervised Learning, Healthcare, UMLS, Natural Language Processing

Sommario

La maggior parte dei documenti clinici sono presenti in forma di dati non strutturati ovvero dati testuali. Tuttavia, per poter applicare dei metodi automatici di apprendimento del contenuto, è necessario fornire una rappresentazione matematica del testo; questa è una delle mansioni dell'Elaborazione del Linguaggio Naturale. In questo lavoro di tesi sarà trattato il dataset E3C, che contiene 10473 casi clinici, con l'obiettivo di produrre dei clusters di documenti, utili al personale medico. Il dataset non fornisce etichette per i documenti, pertanto è stato necessario un approccio non supervisionato. Ispirati dal recente successo delle Reti Neurali a Grafo (GNNs) nel simile ambito di classificazione supervisionata di testi, abbiamo prodotto una modifica totalmente non supervisionata del modello InfoGraph per poter creare delle rappresentazioni vettoriali dei documenti di E3C e tramite dei classici metodi di clustering raggrupparli in base alle loro similarità. I documenti prima di essere stati sottoposti al modello, sono stati rappresentati sotto forma di grafo omogeneo, nel quale ogni nodo rappresenta una parola del testo e i collegamenti fra i nodi rappresentano la connessione fra le parole che appaiono adiacenti nel testo. Per valutare i risultati, in assenza di etichette, abbiamo fatto affidamento sul dizionario UMLS, sul metodo preesistente Doc2Vec e su alcune etichette create in maniera automatica da ChatGPT. Questo è uno dei primi lavori di reti neurali a grafo totalmente non supervisionato, e a nostra conoscenza, il primo che si occupasse di documenti testuali. I risultati mostrano come le rappresentazioni vettoriali dei testi siano in grado di trarre delle informazioni riguardo l'origine dei documenti ma solo in misura minima riguardo il contenuto medico degli stessi. Diverse modifiche possono essere attuate per migliorare le prestazioni del modello, a conferma del fatto che il campo di ricerca in analisi possa offrire molte opportunità di sviluppi futuri.

Parole chiave: Reti Neurali a Grafo, Apprendimento Non Supervisionato, Assistenza Sanitaria, UMLS, Elaborazione del Linguaggio Naturale

Contents

1 | Introduction

1.1. General context

In the last decades, Artificial Intelligence had an incredible performance boost due to the increased interest in the topic and increased computational power of the hardware. Artificial Intelligence is a vast field from which many applications in the real world are present in the everyday lives of all of us. Artificial Intelligence contains other more specific subjects: Statistical Learning, which focuses on causality and correlation between the data and predictions, by using the so-called white-box models, with the goal of providing interpretable results; Machine learning which focuses mainly on predictions; Deep Learning which can be considered a subset of Machine Learning and uses many layers of neural networks to obtain the output, those last two methods are often marked as black-box because of their lack of interpretability.

The goal of this thesis is to develop a method that combines Deep Learning and Statistical Learning to improve an important aspect of the healthcare domain: analyzing medical documents. Hospitals and medical centres produce a large amount of textual documents, related to patient examinations and admissions. These documents contain valuable information about the patients, that are only partially present in more structured data. Their analysis is particularly challenging due to the unstructured nature of this data, which requires to elaborate an appropriate mathematical representation before using them as input of a statistical model.

Given the nature of this problem is possible to go upon literature and see how similar problems have been solved. Indeed one of the most successful areas of deep learning is the so-called Natural Language Processing (NLP), where the general goal is to create models to deal with texts, conversations, words, and sentences, and to make those models more and more similar to our human way of reading, speaking, answering questions.

Medical documents have an additional level of complexity with respect to the other types of documents [18, 27, 66]. While generic texts usually present a limited vocabulary of words, medical documents need to take into account a massive number of highly specific

words representing diseases, body parts, and medicines names; oftentimes, different words refer to the same medical concept. For those reasons, NLP models require more data and training time for medical documents than for other types of documents in order to achieve the same performance. The doctors also use abbreviations and the same clinical case can be described in different ways depending on the doctor, going from an informal description to a more accurate one [23]. Also, the medical documents often present numbers or structured data like tables increasing the variability of the dataset.

Another difficulty to take into account when dealing with medical data is privacy, indeed anonymizing the name of the patient may not be enough to respect the privacy of the person since from the medical description can be retrieved the identity of the patient; this leads to a lack of data available for the training of the models. For non-medical documents, privacy is generally less restrictive so this issue does not jeopardize data availability.

When dealing with machine learning models, an important aspect to consider is the data quality; if there are unbalanced classes for example, the model would be biased to learn more about the features of the biggest class. Therefore is important to provide good-quality data to feed the models, this is often difficult in a medical domain, for the previously mentioned motivations [19].

Many NLP models are focused on supervised tasks, such as classification, but one of the main issues of the medical domain is the lack of labeled data. Labeling medical documents is a very time-consuming process that requires many hours of work by clinicians.

An additional challenge is related to the differences between documents of different countries, not only because of the different languages but also because of the different healthcare systems [47].

Considering the above mentioned challenges, we aim to develop a pipeline for the unsupervised clustering of Italian medical documents. For this purpose, we choose the E3C dataset as an example dataset for our analysis.

This dataset contains text documents of different languages but we focused on Italian because our goal is to implement a tool that can help the Italian healthcare system; also, very few NLP models are trained and tested on the Italian language, and relying on the translation of the text from Italian to English and then apply the model, can lead to a loss of information or some inaccuracies, in addition to being expensive [31]. Therefore focusing on the language of the original text is preferred.

On the other hand, a small portion of the documents belonging to the E3C dataset contains some information on the words, explaining what type of entity some given words

are (e.g. clinical entity, body part). This information can be considered as a sort of pseudo-labeling, still, no information is given to the whole text, and since the goal is to create a tool that given a text is able to understand its content, this forces us to rely on a fully unsupervised method. The lack of labeling also complicates the evaluation part of the model, since no groundtruth can be compared to the results provided by the model.

The two motivations mentioned above are the main difference from the previous research on similar topics, so is difficult to navigate over the pre-existing methods, yet could be an important starting point for future developments.

Summarizing, the goal of this thesis work is to create a pipeline able to translate the documents into a mathematical representation that we will refer to as *document embedding*, i.e. a vector of numbers able to represent the information of the text; and from this mathematical representation deploy some classical clustering methods to evaluate whether similar documents in terms of medical meaning ended up in the same cluster. Having a tool that can group similar documents together can be a great help to medical staff, who can save time in organizing documents by reading them, and can be particularly useful to analyze large amounts of medical documents retrospectively [11]. This pipeline will be applied to the E3C Corpus, a publicly available dataset of clinical documents.

The class of methods that has been chosen in order to create the document embeddings is the so-called Graph Neural Networks (GNNs). This choice takes inspiration from the recent success of GNNs over the similar task of supervised text classification, so the idea developed in this thesis work is to convert the same learning structure into an unsupervised setting. Each document can be converted to a graph which is subsequently elaborated by the GNN model. GNNs models work with a message-passing mechanism where nodes learn from adjacent nodes to create an enhanced representation of themselves and, as a consequence of the whole graph.

In the next section, we will briefly present the structure of this thesis work to enable the reader to navigate more easily through the chapters.

1.2. Thesis Structure

The following chapters have the following structure:

In Chapter 2 there is a deep discussion on GNNs, starting by explaining what a graph is and why it is so useful in representing many real-world problems; and then providing an introduction to GNNs with definitions, presenting the main subclasses of GNNs and the most successful architectures. In this chapter are then analyzed the GNNs models

which are more inherent to our NLP unsupervised problem. A section is dedicated to self-supervised GNNs, which are models used in case of paucity or lack of labels; and another section is dedicated to GNNs tailored for dealing with textual data. Lastly, an overview of non-GNNs methods for NLPs is presented, to provide the reader a smattering of the alternatives to GNNs for tackling similar problems.

In Chapter 3 the European Clinical Case Corpus (E3C) dataset is described. It includes two examples of clinical cases belonging to the E3C dataset, translated into English. The chapter then delves into the data organization of the documents. It also describes the data annotation that some documents are provided by means of the UMLS dictionary. Finally, it offers observations on the various sources of the documents, since E3C collects the clinical cases coming from different origins.

In Chapter 4, the theoretical methodologies of this thesis work are described. The first section concentrates on presenting the developed methodologies for the data preprocessing of the documents, in order to make viable the deployment of machine learning models using textual data. The following section presents the core model of this thesis work: InfoGraph, a self-supervised GNN model that is able to create embeddings given graphs as input. The chapter proceeds explaining the clustering techniques used for grouping similar documents in the same cluster; and the clustering evaluation techniques tailored to estimate, without groundtruth labels, the correctness of the obtained clusters.

In Chapter 5 the results of the embeddings produced by InfoGraph for the E3C dataset are presented, following the methodologies of the previous chapter, analyzing the goodness of the embeddings themselves and their clusterization results. It also contains a description of the performance of the model in terms of execution time by tuning the hyperparameters, and the hardware specification of the computer used for the analysis.

In Chapter 6, the conclusions of this thesis work are presented, summing up the main results and shortcomings, and the suggestions for future development that can be developed in future research.

2 | Graph Neural Networks

In this chapter, the main concepts related to Graph Neural Networks (GNNs) are presented. GNNs are neural networks that are specifically designed to work with data that are in graph form. The key concept is that each node in the graph learns an improved representation of itself from adjacent nodes using aggregation functions that can capture information from neighboring nodes; the weights that measure the importance between nodes are trained by the neural network. It is important to notice that GNNs are a family of models and architectures rather than a single model, so its taxonomy is divided into many sub-families depending on the theoretical background and the application of each architecture.

We think is important to give the reader a solid overview of the literature environment on GNNs before proceeding with our actual case study.

2.1. Graph data structures

Graph-data structures can be explained as a set of objects that are linked by some sort of connection; this connection defines a relation between the objects. Quoting Hamilton's "Graph Representation Learning" Book [21], "The power of the graph formalism lies both in its focus on relationships between points (rather than the properties of individual points), as well as in its generality."

Graphs are a universal language for describing complex systems and some examples of graphs that we can see on a daily basis are:

- Social networks: there are N users subscribed to the system, and a relation between two users can be whether they are mutual friends or not;
- Chemical molecules: a molecule is formed by atoms and those atoms can create chemical bonds with each other;
- Road Maps: is possible to consider cities as objects and the roads that connect them as relations;

Those are just a sub-set of phenomena that can be represented by graphs. We notice that the objects stored in the nodes can be of any type (numbers, vectors, or even non-mathematical entities such as words, people, etc.). In the last decades, the quantity of databases representing graphs has increased exponentially, and, nowadays, the main objective is how to exploit their potential.

Graphs have a strong mathematical foundation developed mainly in the second part of the 20_{th} century. Formally, a graph is defined by the couple $G = (E, V)$, where E is the set of edges and V is the set of nodes (or vertices). The objects are stored as nodes and the relations between them are stored as edges. An edge belonging to E can be written as (i, j), where i and j are the starting and arrival node, respectively.

The most common graph representation is the so-called "Simple Graph", i.e. a graph that has at most one edge between two nodes and does not have self-loops (i.e. an edge connecting a node with itself). An edge $(i, j) \in E$ is called undirected if $(j, i) \in E$. A graph is called undirected if every edge is undirected.

A simple but effective representation for graphs is the so-called Adjacency Matrix, a matrix A in $\mathbb{R}^{v \times v}$ where $v = |V|$, i.e. the cardinality of the set of nodes. $A[i, j] = 1$ if $(i, j) \in E$, $A[i, j] = 0$ otherwise. The matrix A is symmetric in the case of an undirected graph.

The adjacency matrix representation is not optimal in the case of large and non-dense graphs because it would lead to a sparse big matrix A with a few nonnegative entries. An alternative representation is the use of an adjacency list, which notes the edges present in the graph, without the need to store the non-present edges. Some graphs can also have weighted edges, where the entries in the adjacency matrix are arbitrary real-values rather than $\{0, 1\}$, to represent for example probabilities, capacity, etc. More details on graph theory will be introduced later when they will be needed.

2.2. Summary of existing GNNs architectures

Graph datasets, as previously described, have a strong expressive power, and the family of models that have been proven to obtain the best results out of graph datasets are Graph Neural Networks.

GNNs use the graph structure and node features to learn a representation vector of a node or the entire graph. GNNs follow a neighborhood aggregation strategy, where the representation of a node is iteratively updated by aggregating representations of its neighbors. After k iterations of aggregation, a node's representation captures the structural

information within its k-hop network neighborhood. Each node in the graph collects the information of the neighboring nodes through an aggregation function and combines the information aggregated with the representation of the node at the previous time step. The way each node learns from the neighboring nodes is controlled by a learnable weight matrix. Increasing the number of layers allows the information to jump across more edges. In a k layer GNN architecture, the node can get the information of the nodes away at most k nodes and this is called the *receptive field* of the node.

With the graph's geometrical structure and the node information, different learning focuses can be adopted:

- Edge Level: Outputs relate to Edge Classification and Link Prediction;
- Node Level: Outputs relate to Node Classification Node Regression and Node Clustering;
- Graph Level: Outputs relate to Graph Classification and Graph Representation Learning, to obtain a compact representation of graph information.

In Graph Level tasks, after each node has been updated, a readout function is used to summarize the nodes in a single vector containing the representation of the entire graph. To this final vector is possible to add a classification layer that works in the same way as a feed-forward neural network.

We observe that generally in Node Level tasks, the dataset is composed of a unique big graph containing many nodes, while in Graph Level tasks, there are multiple graphs in the dataset and the goal is to learn information about each graph. Multiple publicly available datasets are used as standard benchmarks in this research area. Examples are CITESEER and CORA for the node-level tasks and MUTAG, COLLAB, and PROTEINS for the graph-level tasks.

Furthermore, we observe that in this thesis work we will refer to "Graph Clustering" as the procedure of clustering multiple graphs; in part of the literature instead, this term is used with the meaning of finding clusters of nodes within a graph (e.g. DIFFPOOL [79], MinCutPOOL [3]); we will refer to this last procedure with the term "Node Clustering".

"A comprehensive Survey on Graph Neural Networks" [75] presents the methods developed in the last years on GNNs, proposes a taxonomy of the different architectures and provides a repository of the implementations.

In the following paragraphs, we summarize the taxonomy they proposed.

RecGNNs: a target is learnt by propagating information in an iterative manner until

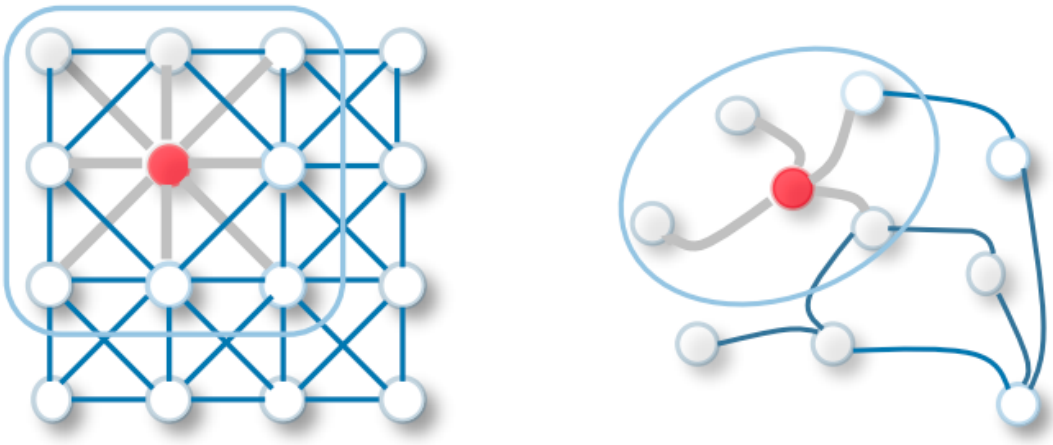


Figure 2.1: In the left 2-D convolution: each pixel in an image is taken as a node where neighbors are determined by filter size. The 2-D convolution takes the weighted average of pixel values of the red node along with its neighbors. The neighboring nodes are ordered and they are in a fixed number. In the right Graph Convolution: to get a hidden representation of the red node, a simple solution is to take the average value of the node features of the red node along with its neighbors. Differently from the image, the neighbours of a node are unordered and variable in size. Picture taken from [75].

.

a stable fixed point is reached. An example of RecGNN is the first GNN, which was developed by (Scarselli et al) [60] in 2008; in this model, a node's hidden state is recurrently updated by:

$$h_v^{(t)} = \sum_{u \in N(v)} f\left(X_v, X^e_{(v,u)}, X_u, h_u^{(t-1)}\right). \tag{2.1}$$

Where $f()$ is a parametric function and $\mathbf{h}_v^{(0)}$ is the embedding of node v at time 0 initialized randomly, $u \in N(v)$ are the neighbouring nodes of v, $\mathbf{X}_v$ the feature vector of node v, $\mathbf{X}_u$ the feature vector of node u, $\mathbf{X}^{\mathbf{e}}_{(v,u)}$ the component (v,u) of the edge feature matrix.

Convolutional GNN: After the success of Convolutional Neural Networks (CNN) [35] in computer vision domain, Convolutional GNN (ConvGNN) started to rise. GNNs are indeed a generalization of CNN: consider a picture, which is composed of pixels; it is possible to connect each pixel to the neighboring ones, and using pixels as vertices and those artificial connections as edges, it is possible to represent a picture with graphs. The learning scheme of CNN is the same of ConvGNN. See Figure 2.1 taken from [75] for a visual example.

ConvGNN architectures are divided into **Spectral Based Approaches** and **Spatial Based Approaches**.

Spectral based methods have a solid mathematical foundation in signal processing. The normalized graph Laplacian matrix is a mathematical representation of an undirected graph, defined as $L = I_n - D^{-(1/2)}AD^{-(1/2)}$, where $D = \sum_j A_{i,j}$ is a diagonal matrix of node degrees. The normalized graph Laplacian matrix possesses the property of being real symmetric positive semidefinite. With this property, the normalized Laplacian matrix can be factored as $L = U\Lambda U^T$, where $U \in \mathbb{R}^{n\times n}$ is the matrix of eigenvectors ordered by eigenvalues and Λ is the diagonal matrix of eigenvalues (spectrum). The eigenvectors of the normalized Laplacian matrix form an orthonormal space, in mathematical words $U\Lambda U^T = I$.

The graph convolution of the input signal x with a filter $g \in \mathbb{R}^n$ is defined as:

$$x *_G g_\theta = U g_\theta U^T x. \tag{2.2}$$

Where $g_\theta = diag(U^T g)$. All spectral based ConvGNN follow the same definition, the only difference is the choice of the graph filter g_θ.

The most famous example of spectral based ConvGNN is GCN (Graph Convolutional Networks) by Kipf and Welling [30], where with a linear approximation of Equation 2.2 they reached the following formula:

$$x *_G g_\theta = \theta(I_n + D^{-1/2}AD^{-1/2})x. \tag{2.3}$$

The final result, generalizing to a signal $X \in \mathbb{R}^{N\times C}$ with C the dimension of the feature vector associated with every node, N the number of nodes in the graph and F filters of the feature map is:

$$Z = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}X\Theta. \tag{2.4}$$

Where $\tilde{A} = A+I_N$, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, a renormalization trick used to avoid vanishing/exploding gradient, $\Theta \in R^{C\times F}$ is a matrix of parameters and $Z \in R^{N\times F}$ is the convolved signal matrix.

The embedding for each node is therefore:

$$h_i^{(t+1)} = ReLU\left(\Theta \frac{1}{\sqrt{deg(v_i)}\sqrt{deg(v_j)}} \sum_{j\in N_i} h_i^{(t)}\right). \tag{2.5}$$

Where ReLU is an activation function defined as $ReLU(x) = max\{0, x\}$ and $deg(v_i)$ is the number of edges connected to v_i. And $h_i^{(0)} = X_i$ i.e. the node's hidden state is initialized with the value of the feature vector of the same node.

In spatial based methods, convolution is based on the node's spatial relations, in which graph convolutions convolve the central node's representation with its neighbors' representations to derive the updated representation for the central node; essentially graph convolutional operation propagates node information along edges.

The difference between spatial based ConvGNN and RecGNN is that in ConvGNN there is a limited number of layers with different parameters per layer, while RecGNN stops learning when a fixed point is reached. Famous examples of spatial based methods are GAT by Velickovic et al. [69] and GIN by Xu et al. [76].

A difference between a spectral based method like GCN and a spatial based method like GAT is that GCN assigns a non-parametric weight $a_{ij} = \frac{1}{\sqrt{deg(v_i)}\sqrt{deg(v_j)}}$, while GAT captures the weight a_{ij} via neural network attention. So GAT is parametric while GCN is parametric only over the features.

GIN is one of the most famous spatial based methods, its idea is based on the Weisfeiler-Lehman (WL) test (1968) [72], which tries to solve the *Graph Isomorphism Problem* i.e. understanding whether two graphs are topologically identical. Apart from a few, specific cases, the WL test is able to distinguish a broad class of graphs. In [76] the following theorem is stated:

Theorem 2.1. *Let $G1$ and $G2$ be any two non-isomorphic graphs. If a graph neural network $A : G \to R^d$ maps $G1$ and $G2$ to two different embeddings, the Weisfeiler-Lehman graph isomorphism test also decides $G1$ and $G2$ are not isomorphic.*

In other words, any aggregation-based GNN is at most powerful as the WL test in distinguishing different graphs. The authors reached in a formal way to the conclusion that there is a whole class of GNNs that are as powerful as the WL test. In this class, the model they proposed, called GIN, adopts the following node update rule:

$$h_v^{(k)} = \mathbf{MLP}^{(k)}\left((1 + \epsilon^{(k)})h_v^{(k-1)} + \sum_{u \in N_v} h_u^{(k-1)}\right). \tag{2.6}$$

Where ϵ is a fixed scalar or a learnable parameter. Furthermore, the authors of GIN, proved that *sum* aggregator has the best expressive power, compared to *mean* and *max*. The expressive power of a graph-based model is its capability to distinguish two noniso-

morphic (i.e. topologically different) graphs. In conclusion, GIN with *sum* aggregator is the best choice in terms of expressive power, this is reflected in the state of the art performance of GIN for graph classification tasks. Nevertheless, is important to notice that for example, *mean* aggregator works well when statistical information is more important than the exact structure for the task, this happens for example in node-level tasks. For this reason, GCN [30] and other non-powerful GNNs outperform GIN in node classification tasks.

Graph Autoencoders (GAEs): GAEs are deep neural networks architectures that map nodes into a latent feature space and decode graph information from latent representation. GAEs can be used to learn network embeddings or generate new graphs. A successful example is GAE by Kipf and Welling [29] which leverages GCN [30] to encode node structural information and node feature information by reconstructing the adjacency matrix A with the goal of clustering the nodes of a graph. They also proposed VGAE which adds a variational probabilistic component to the same model. Other examples of this class are ARVGA (Adversarially Regularized Graph Autoencoder for Graph Embedding) [49] and MGAE (Marginalized Graph Autoencoder) [71].

Spatial Temporal Graph Neural Networks: this family of GNNs was developed to capture the time dependency that some phenomena have and that can also be represented well by graphs. For example the traffic in a road network in the different moments of a day. An example is ST-GCN [77].

2.3. Task-based layers

As previously described, GNNs are suitable for solving different types of problems, focusing on either node, edge, or graph level. For every GNN model presented before, the result is a refined version of the input nodes representation. From this representation, depending on the task of interest a suitable final layer should be applied, in order to apply a loss function to train the network parameters. Following the survey proposed by Dwivedi et al. [16] we present three supervised task-based layers:

Graph classifier layer: To perform graph classification, we first build a d-dimensional graph-level vector representation y_G by averaging over all node features in the final GCN layer:

$$y_G = \frac{1}{V}\sum_{i=0}^{V} h_i^L. \tag{2.7}$$

The graph-level vector representation is then passed to a MLP, which outputs logits/scores $y_{pred} \in \mathbb{R}^C$ for each class:

$$y_{pred} = P\mathbf{ReLU}(Qy_G). \tag{2.8}$$

Where $P \in \mathbb{R}^{d \times C}$, $Q \in \mathbb{R}^{d \times d}$ and C is the number of classes. Finally, we minimize the cross-entropy loss between the logits and groundtruth labels:

$$CrossEntropy(y, y_{pred}) = -\frac{1}{V}\sum_{i=0}^{V} y_i \log(y_{pred_i}). \tag{2.9}$$

Graph regression layer: To perform graph regression, we compute y_G using equation 2.7 and pass it to an MLP which gives the prediction score $y_{pred} \in \mathbb{R}$

$$y_{pred} = P\mathbf{ReLU}(Qy_G). \tag{2.10}$$

Where $P \in \mathbb{R}^{d \times 1}$, $Q \in \mathbb{R}^{d \times d}$. The L1-loss between the predicted score and the groundtruth score is minimized during the training:

$$L1(y, y_{pred}) = \frac{1}{V}\sum_{i=0}^{V} \|y_i - (y_{pred_i})\|. \tag{2.11}$$

Node classifier layer: To perform node classification, we independently pass each node's feature vector to an MLP for computing the logits $y_{i,pred} \in \mathbb{R}^C$ for each class:

$$y_{i,pred} = P\mathbf{ReLU}(Qh_i^L). \tag{2.12}$$

Where $P \in \mathbb{R}^{d \times c}$, $Q \in \mathbb{R}^{d \times d}$. The cross-entropy loss weighted inversely by the class size can be used during training.

2.4. Self-supervised learning for graphs

Most of the successful accomplishments of neural-network-based models came from a supervised setting, where the model is trained by minimizing the error function given by the prediction of the model and the groundtruth. In this thesis we work in a complete absence of labels during training, this is an important aspect that will be discussed from

several perspectives throughout the chapters. In particular *self-supervised learning* is one of the possible solutions to the problem of labels absence, and this technique will be adopted as a practical solution to the text clustering problem in the following chapters.

Self-supervised learning is the learning strategy that uses the input data itself to make the model learn in a supervised-like way, without relying on externally provided labels. The model learns to predict parts of the input, and those predictions are used as an artificial method to obtain a loss function to minimize and then train the model.

When self-supervised learning is adopted in combination with labels, the final results can benefit of both learning schemas: the pre-training part is done in a self-supervised way to learn the weights of the neural networks, and then a fine-tuning part is done using supervised learning to increase the accuracy by refining those weights. Two famous NLP models that used the pre-training and fine-tuning approach are GPT [55] and BERT [13].

Nevertheless, self-supervised learning can be used without any kind of label. Self-supervised models can be divided into two classes:

- *Autoassociative self-supervised learning*: where a neural network is trained to reproduce the input. An example are autoencoder models (e.g. [24]), which are able to produce a compact representation of the input vector by means of minimizing the reconstruction loss, i.e. reconstruction error between the true input and the reconstructed one;
- *Contrastive self-supervised learning*: where data augmentation is applied to the input and the learning schema is to minimize the difference between positive instances (augmented ones) and maximize the distance between negative instances (similar data which are not taken from the original input). An example is [36] where contrastive learning is used to cluster images.

In recent years self-supervised learning has been tailored to work on graph datasets. In the next paragraphs, we will follow "Graph Self-Supervised Learning: A Survey" by Liu et al. [39] to present the taxonomy, the important results, and the issues of self-supervised on graphs.

We first start with some important definitions. *Downstream tasks* are the graph analysis tasks used to evaluate the quality of feature representation learned from different models. Typical applications are node classification, graph classification, and graph clustering. *Pre-text tasks* refer to predefined tasks for models to solve (e.g. graph reconstruction), which help models learn more generalized representations from unlabeled data. Those tasks are auxiliary tasks in the sense that our goal is not to solve them, but they are

useful to enhance downstream tasks. Graph self-supervised learning can be divided into four types, including generation-based, auxiliary property-based, contrastive-based, and hybrid methods.

Generation-based methods can both learn to reconstruct the graph's feature matrix information or graph's topological information, namely adjacency matrix. The former learns more on node-level knowledge, and the latter on node pair-level knowledge. Graph Completion [80], Attribute Mask [28] and GAE [29] are examples of this class.

Auxiliary property-based methods use graph properties as pseudo labels, e.g. closeness centrality and degree centrality. To learn a useful representation of the graph.

Contrastive-based methods are built on the idea of Mutual Information (MI) maximization between two augmented instances; the augmented instances can be obtained by node feature masking or shuffling, edge modification, graph diffusion, etc. Examples of methods for this class are InfoGraph [64] and GraphCL [20].

Hybrid methods exploit the previous models' virtues combining them and trying to filter the shortcomings.

Is important to notice that several learning schemas can be adopted during training: *Pre-training and Fine-tuning (PF)*, *Joint Learning (JL)*, and *Unsupervised Representation Learning (URL)*. In (PF) the model is first pre-trained with pretext tasks on pre-training datasets, which can be viewed as an initialization for the encoder's parameters. After that, the pre-trained model is fine-tuned with fine-tuning datasets (with labels). In (JL) scheme, the model is jointly trained with the pretext and downstream tasks. In (URL) instead, no labels are used during training, the first part is similar to (PF) but after pre-training the model parameters are frozen and then passed to the downstream task (which can be classification, regression, clustering, etc.). With those information is clear that the only training schema that we can use for our practical application is (URL), given the absence of labels. Furthermore, almost all (URL) methods of the survey, after the unsupervised training part, used a supervised downstream task with labels (supervised learning), this is something that we cannot do, so we will replace the last part with an unsupervised downstream task, as will be shown in Chapter 4.

2.5. GNNs for NLP

In the previous sections, we presented the power of graph-based representation and GNNs in a general framework. In this section, we will focus on the specific GNN application of this thesis: Natural Language Processing (NLP). We already mentioned that graphs

can represent a rich variety of information and, in particular, text data is well suited for a graph representation, as we will explain in the following paragraphs. To introduce the reasons and the different methodologies of representing texts as graphs, we will follow the survey by Sonowane et al. [62].

Nowadays text is the most common form of storing information. The representation of documents is an important step in the process of text mining. Hence, the challenging task is the appropriate representation of the textual information which will be capable of representing the semantic information of the text.

Graph representation can model relationships and structural information effectively. A text can appropriately be represented as a graph using vertices as feature terms, which can be words, sentences, paragraphs etc., and edges as relations between the feature terms, which can be syntactical, semantical, or proximity relations. Syntactical relations between words are the grammatical dependencies such as subject, object, and predicate verb, they determine how the words are linked in the sentence. Semantical relations between words are the meaning and the interpretation that related words can have, they go beyond the grammatical meaning and can represent relationships such as cause-effect, spatial-temporal and so on. Proximity relations consider the spatial proximity of words within the text. Words that appear close together in a sentence or document are likely related in some way, both syntactically and semantically. Proximity-based representations can capture local contexts and co-occurrence patterns, which are valuable for tasks such as context-aware analysis.

An example of graph representation of textual sentences is shown in Figure 2.2 taken from [42]. Here the objective is Semantic Role Labelling, i.e. assigning labels to the various arguments of a verb, such as subject, object, agent, location, time, etc. This example shows how well a graph can explain the semantical and syntactical relations of a sentence. An example that shows instead the capability of graphs in describing proximity relations between words in a text is shown in Figure 2.3 taken from "Graph neural networks for natural language processing: A survey" [73].

Graph model is the most suitable representation of text document; furthermore, it does not require detailed linguistic knowledge, domain or language specific collection. Graph-based representations excel at capturing complex relationships and interactions between text elements. By modeling syntactic, semantic, and proximity-based relations, graphs can encapsulate rich contextual information about the text. The application of graph-based representation of text elements provides processing of the information in various areas like document clustering and document classification.

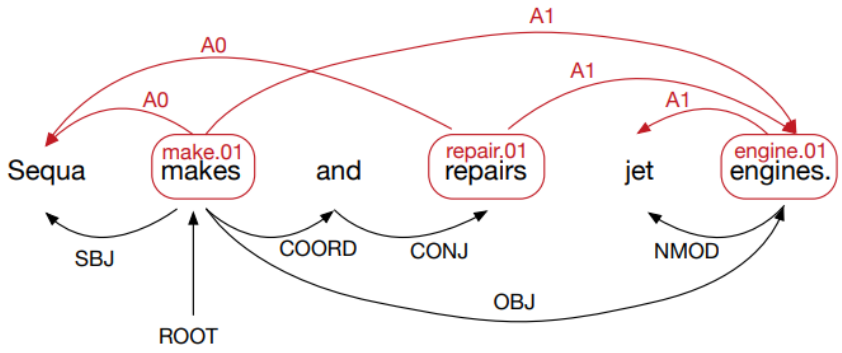


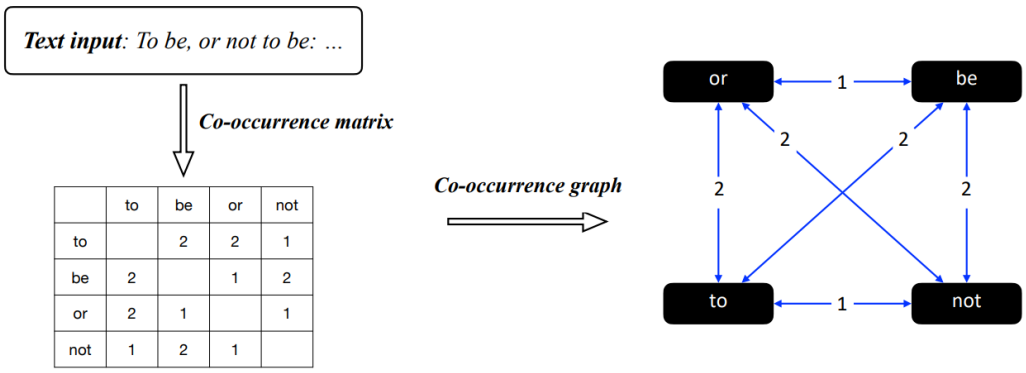
Figure 2.2: In the top of the sentence, the edges represent semantical dependencies between words, below the sentence the edges represent syntactical dependencies between words. Each word is considered as a vertex of the graph which is the whole sentence. Picture taken from [42].

.

Given this introduction on text representation using graphs, it appears that graphs are a good choice that we have for automatic document analysis and to represent textual information in general. In the following paragraphs we will present some researches that used graphs for describing text and in addition, used GNNs-based models to extract information from the text. In particular, we will present methods developed for supervised text classification.

Yao et al. [78], developed TextGCN, a large and heterogeneous graph for text classification, which contains word nodes and document nodes so that global word co-occurrence can be explicitly modeled: the same word occurring in different documents represent the same node in the graph. The number of nodes in the text graph is the number of documents (corpus size) plus the number of unique words. So, there is a unique graph containing all the corpus and all the vocabulary. This was a pioneering idea that proved state-of-the-art results on text classification benchmarks but with the drawback of a transductive setting, meaning that the model cannot make classifications on unseen documents, since the training phase uses both training and test documents to learn the parameters of the model.

To overcome the problem, Huang et al. [26] proposed textLevelGCN, which is a GNN-based model that builds graphs for each input text with global parameters sharing instead of a single graph for the whole corpus. Each graph, associated with each text, is constructed considering nodes as words and for each word, the connections with the other words depend on the proximity of the two.



	to	be	or	not
to		2	2	1
be	2		1	2
or	2	1		1
not	1	2	1	

Figure 2.3: In the right, the graph representation of the sentence is provided, in the left, the corresponding co-occurrence matrix, where each entry shows the frequency in which two words appeared in the same window size (in this example the window size is equal to 3) Picture taken from [73].

.

Other valuable graph structure ideas were developed by Ding et al. [14] with HyperGCN, a generalization of the graph representation power to hypergraphs; and by Linmei et al. [37], who created a different type of heterogeneous graph, containing entities, documents and topics as nodes.

In conclusion, text representation using a graph model provides various operations that are helpful in many applications of information retrieval. In this setting, while supervised and semisupervised approaches are deeply studied, unsupervised approaches are almost neglected. In Chapter 4, we will show the type of transformation from text to graph and the unsupervised model that we adopted to tackle the document clustering objective of this thesis work.

2.6. Existing non-GNNs-based methods for NLP

As discussed in the previous section, GNNs and graph structures proved to excel in NLP tasks such as text classification, nevertheless, other techniques have been developed to tackle the same tasks using different approaches from graphs. In this section, we present the main methods that distinguished themself from the others and that have been applied to a rich variety of NLP tasks throughout the last decades.

Traditional models like *Vector Space Models* consider numerical feature vectors in an Euclidean space. Each term present in the corpus is a feature of the model, therefore documents are represented by weights given to each word present within the documents

computed according to their importance. The most common method to assign weights to the terms is TF-IDF (i.e. Term Frequency - Inverse Document Frequency) [59], which is a measure of the importance of the term t for the document d, given by the product of TF and IDF. TF measures the frequency of a term t within the document d, following the idea that the weight of a recurring term in a document is proportional to the frequency of the term itself (Formula 2.13). IDF [63] instead, assigns the weights considering the fact that some nonimportant words appear very often in all the documents and therefore TF may not be the best solution, so the weight of IDF measures the specificity of a term as an inverse function of the number of documents in which it is occurring (Formula 2.14, D is the set of documents). TF-IDF is the product of TF and IDF to have a trade-off between the specificity and the frequency of a term within the corpus, its final representation is shown in Formula 2.15. With the vectorial form of the documents is then possible to rank them according to their similarity using measures such as cosine similarity.

$$\mathrm{TF}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}. \tag{2.13}$$

$$\mathrm{IDF}(t,d) = \frac{|D|}{|\{d \in D : t \in d\}|}. \tag{2.14}$$

$$\text{TF-IDF}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \times \frac{|D|}{|\{d \in D : t \in d\}|}. \tag{2.15}$$

The vector space model has the following disadvantages:

- As previously explained, each term present in the corpus is a feature of the model so the Euclidean space of the document embeddings has enormous dimensionality (thousands of features), leading to the problem of curse of dimensionality;
- Each word representation is independent from the other; thus, word appearance ordering or other relations cannot be represented;
- If two documents have similar meanings but are composed of different words, similarity cannot be computed easily.

Another valuable option to deal with text data are Recurrent Neural Networks (RNNs) [58] and their advanced improved architectures such as LSTM [25]. The key component of the RNNs family of models is that they allow to process the input across different time steps while keeping information of the past inputs. This makes them suitable to work with sequential data, where inputs are given one after the other. Text data is

an example of sequential data, and the words are the sequential inputs, indeed, many applications of RNNS to NLP tasks have been developed [65] [9] [38]. On the other hand, RNNs models struggle to capture long-term dependencies due to the vanishing gradient problem. GNNs are preferred over RNNs because GNNs can capture both local and global dependencies between structural components. Therefore, they can capture rich semantic relationships and dependencies that are important for the task. Additionally, unlike many sequence models, GNNs can naturally handle variable-length inputs by operating on the graph structure, without any need to map every data sample to a fixed-sized vector. Furthermore, while longer documents can be particularly challenging for RNNs, GNNs methods hold particular promise for longer documents, as explained by Bugueno et al. [8].

A new architecture, that emerged more recently is *Transformer* by Vaswani et al. [68]. It was originally designed to tackle the problem of neural machine translation using an encoder-decoder structure with an attention mechanism (the transformer architecture can be seen in Figure 2.4). Rapidly, transformers have also been deployed to different tasks other than just neural machine translation, achieving state of the art results. Transformers, like RNNs, work on sequential data such as text, but differently from RNNs, they process the data simultaneously reducing the problem of vanishing gradient. In order to process the data simultaneously and still retain the order information, positional encodings are used. Many positional encodings can be chosen but the choice of the authors came down to sine and cosine functions:

$$PE(pos, 2i) = sin(pos/10000^{2i/d_{model}}). \tag{2.16}$$

$$PE(pos, 2i+1) = cos(pos/10000^{2i/d_{model}}). \tag{2.17}$$

The multi head attention used in the transformer architecture is a tool to analyze complex sequences of tokens, allowing each head to focus on different parts of the sentence, increasing the robustness of the results.

A rich variety of models have been developed based on transformers, for multiple and different types of tasks; two examples are the already mentioned, GPT [55] and BERT [13]. Furthermore, BERT embeddings have also been used in clinical settings, improving the results over many NLP clinical tasks [2].

The work by Bugueno et al. [8] shows that although the effectiveness of graphs depends on the textual input features and domain, simple graph constructions perform better the longer the documents, and graph representations are especially beneficial for longer

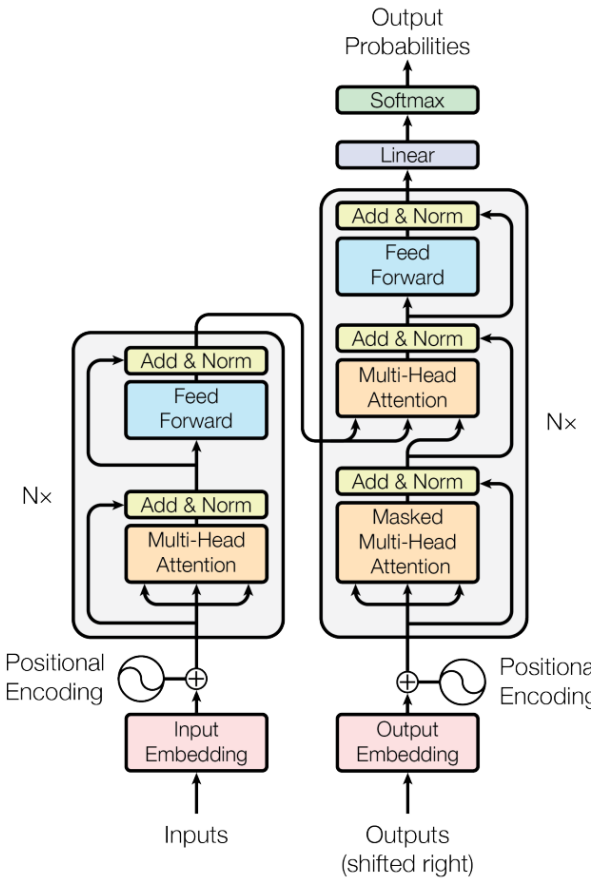


Figure 2.4: Transformer architecture. The left part is the encoder and the right part is the decoder. Inputs are processed simultaneously. Picture taken from Transformer original paper [68].

.

documents, outperforming transformer-based models. This is firstly because the original BERT model for example has a limit over the number of tokens to process in a text (512 tokens), while GNN-based methods don't have any limit; and secondly because even not taking into consideration this limit, BERT model perform better in shorter documents. Considering that documents in our dataset have lengths that can vary remarkably, GNN-based methods appear to be a better solution than transformer-based methods.

Lastly, Dwivedi and Bresson [15] propose a link between GNNs and transformers, indeed the words processed by transformers are in the form of fully connected dense graphs, where each word is connected to every other word in the text. By mixing GNNs' power to process every type of input graph and the transformer's power to understand the context of the words, the authors developed a generalization of transformer architecture that takes into account the inductive bias of the graph created from the text, leading to a

more efficient architecture than the original transformer.

In conclusion, even though many models have shown incredible results for NLP tasks, graph neural networks may outperform the methods mentioned above, since GNNs can capture better the connectivity between the terms and also because GNNs can be seen as a generalization of the previous methods.

3 | Dataset

3.1. E3C-Corpus

The European Clinical Case Corpus (E3C) [41], is a freely available multilingual corpus (English, French, Italian, Spanish, and Basque) of semantically annotated clinical narratives to allow for the linguistic analysis, benchmarking, and training of information extraction systems[1].

In this work, only the Italian sub-corpus will be considered, since our goal is to provide a model that would work on Italian texts, specifically. The motivation for which we decided to work only on Italian texts is that language models are generally trained and tested on English texts; this leads to a structural bias when we apply the models to another language, indeed Italian and English grammar and syntax are different in many aspects, so models that work well in English may not work well in Italian. Using Italian documents for building our models should allow for better outcomes [47].

Each document of E3C describes a clinical case of a patient i.e. a statement of a clinical practice, presenting the reason for a clinical visit, the description of physical exams, the assessment of the patient's situation, and in some cases, his medical history.

We present two examples of clinical cases taken from the dataset:

Example 1:

"Un uomo di 27 anni si presenta alla nostra attenzione per insorgenza improvvisa di dolore a livello dell'emitorace destro irradiato alla spalla. In anamnesi riferiti disturbi dell'alvo con feci poco formate e occasionale riscontro di muco e sangue da un anno. Agli esami ematochimici: incremento di PCR e VES, severa anemia microcitica ipocromica (Hb 6,5 g/dl), leucocitosi neutrofila, piastrinosi, aumento del D-dimero, ipoferritinemia. All'esame obiettivo del torace: MV ridotto al campo medio di destra, praticamente abolito in sede basale bilateralmente; all' EGA alcalosi respiratoria; all'ECG tachicardia sinusale

[1] `https://github.com/hltfbk/E3C-Corpus`

con blocco incompleto di branca destra. Veniva pertanto effettuata TC torace che mostrava quadro di embolia polmonare. Negativo il doppler artero-venoso degli arti inferiori. Ecocardiografia nei limiti. Per positività del sangue occulto fecale e della calprotectina fecale veniva praticata EGDS risultata negativa e colonscopia che diagnosticava rettocolite ulcerosa acuta estesa a tutto il colon. Uno screening per patologie trombofiliche risultava positivo per mutazione in eterozigosi del gene dell'enzima MTHFR e in omozigosi del polimorfismo 5G/4G del promotore del gene PAI. In relazione al riscontro di elevati valori di anticorpi anti-muscolo liscio, nel sospetto di colangite sclerosante primitiva, veniva inoltre effettuata Colangio-RM risultata negativa. ll paziente veniva sottoposto ad emotrasfusione, a terapia con warfarin, mesalazina e ferro. Le condizioni generali sono progressivamente migliorate con stabilizzazione dei valori di emoglobina ed il paziente è stato pertanto dimesso. Ad un mese dalla dimissione nella norma i valori di emoglobina con normalizzazione della piastrinosi. Continua tutt'ora la terapia con mesalazina e con warfarin."

The translation of the previous Italian text is[2]:

"A 27-year-old man comes to our attention for sudden onset of pain in the right hemithorax radiating to the shoulder. In history reported alvus disturbances with poorly formed stools and occasional occurrence of mucus and blood since one year. On hematochemical examination: increased CRP and ESR, severe hypochromic microcytic anemia (Hb 6.5 g/dl), neutrophilic leukocytosis, plateletosis, increased D-dimer, hypoferritinemia. On objective chest examination: MV reduced at right midfield, practically abolished at basal bilaterally; on EGA respiratory alkalosis; on ECG sinus tachycardia with incomplete right bundle branch block. Chest CT was therefore performed, which showed picture of pulmonary embolism. Negative lower extremity arteriovenous doppler. Echocardiography within limits. For fecal occult blood and fecal calprotectin positivity, EGDS was performed which was negative and colonoscopy diagnosing acute ulcerative rectocolitis extended to the whole colon. A screening for thrombophilic disorders was positive for mutation in heterozygosity of the MTHFR enzyme gene and in homozygosity of the 5G/4G polymorphism of the PAI gene promoter. In relation to the finding of elevated anti-smooth muscle antibody values, in suspicion of primary sclerosing cholangitis, Cholangio-RM was also performed and found to be negative. The patient underwent hemotransfusion, warfarin, mesalazine and iron therapy. The general condition gradually improved with stabilization of hemoglobin values, and the patient was therefore discharged. At one month after dis-

[2]Translations made with `https://www.deepl.com/translator`

charge in normal hemoglobin values with normalization of plateletosis. He still continues mesalazine and warfarin therapy."

Example 2:

"Un paziente lamenta disuria e pollacchiuria. L'analisi delle urine mostra un'ematuria microscopica ma nessuna batteriuria."

The translation of the previous Italian text is:

"A patient complains of dysuria and pollakiuria. Urinalysis shows microscopic hematuria but no bacteriuria."

The first observation that is clear even from just those two examples, is that documents can have different lengths, going from about ten words up to thousands of words. This is an important aspect to keep in mind in the phase of the choice of the model, indeed, some models perform better when the inputs have similar sizes and even some other models require exactly the same input size.

3.2. Data Organization

The documents are divided into three layers:

- Layer 1: Documents in this layer contain two types of annotations: (i) clinical entities: pathologies, symptoms, procedures, body parts, etc., according to standard clinical taxonomies (i.e., UMLS); and (ii) temporal information and factuality: events, time expressions, and temporal relations according to the THYME standard;
- Layer 2: Clinical entities in these documents have been automatically annotated by dictionary matching. The dictionary was obtained by combining the entities present in the Italian UMLS, and the entities annotated in Layer 1;
- Layer 3: non-annotated medical documents to be exploited by semi-supervised or unsupervised approaches.

In the following Table 3.1, is possible to find descriptive statistics of the layers:

	L1	L2	L3
Documents	86	174	10213
Tokens	24319	49900	13601915

Table 3.1: Number of documents and number of tokens per layer

The Italian documents were collected from The Pan African Medical Journal and from existing corpora like the SPACCC corpus. Other documents were collected from admission tests for specialties in medicine, patient information leaflets for medicines, and abstracts of theses in medical science.

On Layer 1 documents, the authors of E3C explain that a basic preprocessing step was applied, removing sentences (at the beginning or at the end of documents) that are not part of clinical cases; removing references to figures and tables; and restoring punctuation and capitalization. For Layer 2 and Layer 3, a different approach for data preprocessing is done for this thesis work and is described in Section 4.1.

3.3. Data Annotation

Layer 1 contains manual annotations, they consist of (i) clinical entities (e.g., pathologies), (ii) temporal information, and (iii) factuality (e.g., events). Layer 2 annotations consist only of clinical entities that have been automatically recognized by dictionary matching. The dictionary of each language was obtained by combining the entities present in the UMLS Italian dictionary, and the entities annotated in Layer 1, so Layer 1 can be considered to be of better quality than Layer 2. Annotations are stored in .xml files. 2 out of 86 documents of Layer 1 don't contain any clinical entity.

Each clinical entity in a document was assigned to one CUI (Concept Unique Identifier) from the UMLS, (Unified Medical Language System) [6]. UMLS is a compendium of many controlled vocabularies of the biomedical sciences originally created in 1986. It provides a mapping structure between these vocabularies and thus enables translation between terminology systems; it can also be viewed as a comprehensive thesaurus and ontology of biomedical concepts. In other words, UMLS is useful because it creates a unique index (CUI) for a set of similar words or synonymous, in order to minimize ambiguity, which often occurs in medical documents.

The UMLS also provides facilities for natural language processing [43] [74]. It is intended to be used primarily by developers of medical informatics systems. The motivation of

UMLS lies in the fact that the number of biomedical resources available to researchers is enormous. This is often a problem due to the large volume of documents retrieved when searching the medical literature. The purpose of the UMLS is to improve access to this literature by facilitating the development of computer systems capable of understanding biomedical language [6].

Data annotations and in particular clinical entities will be used as an attempt to evaluate the document embeddings produced by our clustering pipeline as will be discussed in depth in Chapter 4.

Examples of clinical entities annotated are reported in Table 3.2. Those clinical entities are taken from the previous example of a clinical document that belongs to layer 1 and therefore clinical entities have been annotated.

Clinical Entity	CUI term	CUI index
Dolore a livello dell'emitorace destro	Right hemithoracic chest pain	C2073322
Anemia microcitica ipocromica	Microcytic hypochromic anemia	C0271901
Leucocitosi neutrofila	Leukocytosis	C0023518
Piastrinosi	Thrombocytosis	C0836924
Ipoferritinemia	Hypoferritinemia	C5243686
Alcalosi respiratoria	Alkalosis, Respiratory	C0002064
Tachicardia sinusale	Sinus Tachycardia	C0039239
Embolia polmonare	Pulmonary Embolism	C0034065
Rettocolite ulcerosa acuta	Acute ulcerative rectosigmoiditis	C5437694
Colangite sclerosante primitiva	Primary sclerosing cholangitis	C0566602

Table 3.2: Clinical entities referring to **Example 1** text presented some paragraphs before and taken from layer 1. The CUI terms and the CUI index are the translations according to UMLS of the clinical entities occurring in the text.

3.4. Sources

In this section, we want to analyze the differences among the documents of the corpus with respect to the different sources and the different layers to which they belong. Is important to be aware of those peculiarities within the data because we cannot assume that each text as input is independent and identically distributed, since two texts belonging to different sources may differ in many aspects for example in the text length, in the medical topic

of discussion and in the form in which the text is written. In Table 3.3, Table 3.4 and Table 3.5, for each layer, we summarize the descriptive statistics of the set of documents grouped by source, presenting the number of documents with the same source, their average length, their standard deviation of the length and their minimum and maximum length. With the length of a document, here we refer to the length of the string which represents the text of a document. In Figure 3.1, Figure 3.2 and Figure 3.3 are shown the box-plots of the length of the documents belonging to layer 1, layer 2 and layer 3, respectively.

From the tables and form the box-plots, we want to highlight some important details about the data:

- Some sources present documents on a specific topic only, for example, "Pediatric Reports" and "GCND Giornale di Clinica Nefrologica e Dialisi" while other sources present documents whose topic can span across all the fields of medicine, for example, "AboutOpen" and "Working Paper of Public Health";

- The documents belonging to the sources "Sapienza Università di Roma" and "Miur" have an average length that is much lower than the documents belonging to other sources. The reason for this is that those documents are describing case studies for medical or nursing university exams. The documents belonging to the other sources instead are journal publications that describe in depth the clinical case of the patient, so for this reason, the text is longer. See for example Figure 3.1 for a visual explanation of the different lengths of the documents grouped by source, for layer 1.

- In layer 3 only, are present documents from "AIFA" ("Agenzia Italiana del Farmaco"), constituting the large majority of layer 3 itself. Those documents represent medicine leaflets, for this reason, they are the longest on average 3.3. Furthermore, no information about clinical cases is reported in those documents, so we will discuss whether to keep those documents or not in Chapter 4, in order to see how their presence is useful for our analysis.

- Analyzing the values of min, max, mean, and standard deviation of the length of the text for each group of documents with the same source, we can conclude that there is a homogeneity in the length of the documents belonging to the same group. Furthermore, no outliers have been detected based on the length of the texts, so we will keep all the documents for the analysis.

- Documents with "Pediatric Reports" as a source have been split according to the

Source	Docs	Avg Length	Std Dev	Min	Max
AboutOpen	2	3267	619.4	2829	3705
Giornale di Clinica Nefrologica e Dialisi	2	3116	384.6	2844	3388
Argomenti di Oncologia Geriatrica	1	3829	0	3829	3829
Miur	4	623.5	57.5	579	708
Clinical Management Issues	5	3673.2	351.7	3106	4043
Italian Journal of Medicine	33	1468.3	749.1	598	3210
Italian Journal of Emergency Medicine	3	2923.3	1095.2	1661	3622
Sapienza Università di Roma	11	668	45.1	596	718
Microbiologia Medica	1	1716	933.2	1716	1716
La Pediatria Medica e Chirurgica	7	2466.8	678.7	1834	3410
Pediatric Reports (PAGEPresse)	8	2716.8	636.8	1963	3438
Pediatric Reports (MDPI)	6	3127.8	169.0	2975	3368
Prof. Dr. Francescopaolo Mattioli	3	3177	609.8	2511	3708

Table 3.3: Descriptive statistics of the documents grouped by source for layer 1.

different publishers of each document (MDPI and PAGEPress).

Source	Docs	Avg Length	Std Dev	Min	Max
AboutOpen	2	3204.5	767.2	2662	3747
Giornale di Clinica Nefrologica e Dialisi	10	3384.3	803.9	1770	4057
Miur	12	655.3	49.8	605	739
Clinical Management Issues	7	3922.5	519.6	3088	4531
Hematology Reports	1	2631	0	2631	2631
Italian Journal of Medicine	70	1461	742.3	635	3373
Italian Journal of Emergency Medicine	8	2652.7	1087.8	663	4020
Sapienza Università di Roma	15	657.7	65.3	544	745
Microbiologia Medica	9	2654.3	933.2	1659	4332
La Pediatria Medica e Chirurgica	6	2400.5	604.4	1685	3350
Pediatric Reports (PAGEPresse)	7	2439.1	709.9	1671	3494
Pediatric Reports (MDPI)	16	3173.9	330.7	2469	3637
Working Paper of Public Health	2	2507	794.7	1945	3069
Prof. Dr. Francescopaolo Mattioli	9	3671	647.6	2048	4263

Table 3.4: Descriptive statistics of the documents grouped by source for layer 2.

Source	Docs	Avg Length	Std Dev	Min	Max
Giornale di Clinica Nefrologica e Dialisi	10	4698.4	1017	2745	6579
Miur	1439	230.3	129.3	38	915
Clinical Management Issues	1	4463	0	4463	4463
Hematology Reports	6	1143.8	328.5	789	1664
Italian Journal of Medicine	534	1170.9	300.3	364	2850
Italian Journal of Emergency Medicine	4	2760.2	2161.8	420	5241
Sapienza Università di Roma	89	598.5	235.6	197	1287
Microbiologia Medica	17	1405.7	1129.1	535	5593
La Pediatria Medica e Chirurgica	6	3104.5	2288.5	428	5842
Pediatric Reports (PAGEPresse)	8	3465.3	1834.3	1334	6096
Pediatric Reports (MDPI)	6	3117.6	314.6	2658	3439
Prof. Dr. Francescopaolo Mattioli	9	6055.1	2817.8	4213	13387
Agenzia Italiana del Farmaco	8084	10373.4	4775.6	2074	58013

Table 3.5: Descriptive statistics of the documents grouped by source for layer 3.

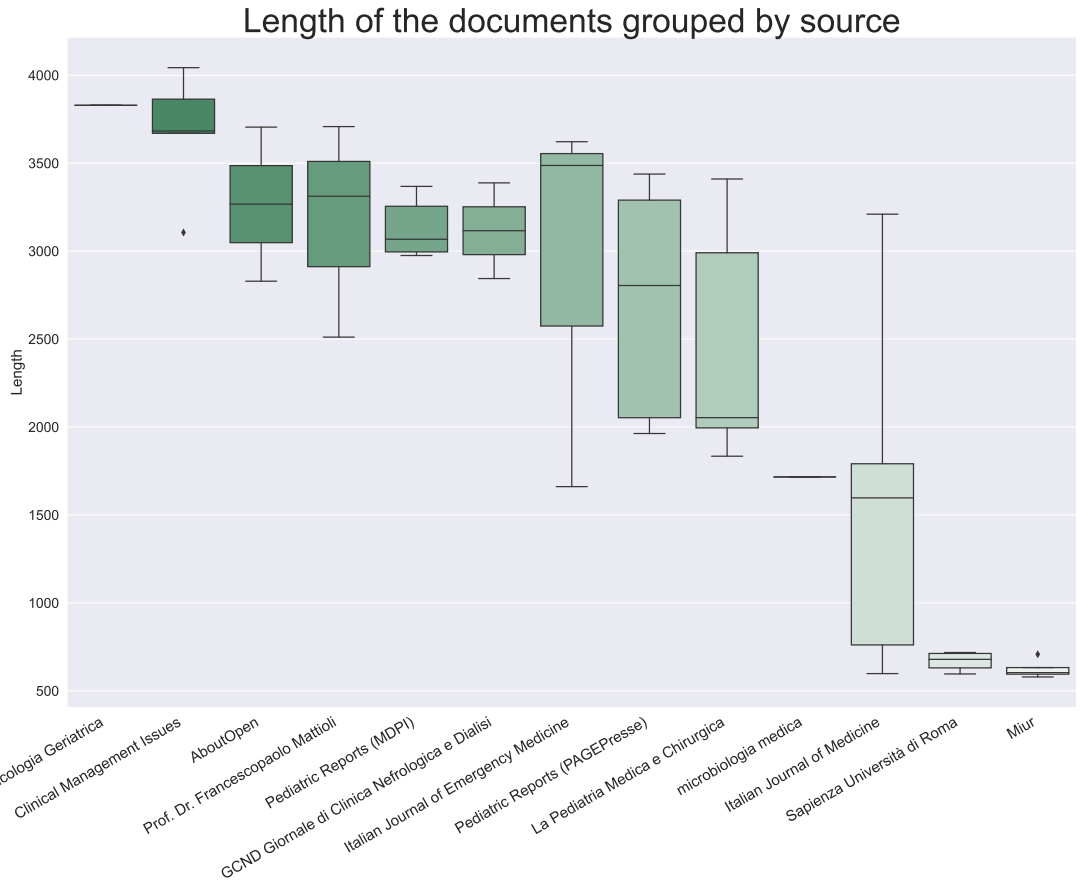


Figure 3.1: Boxplot showing the length of the documents grouped by source for layer 1. It is evident that "Miur" and "Sapienza Università di Roma" are the sources that correspond to shorter documents.

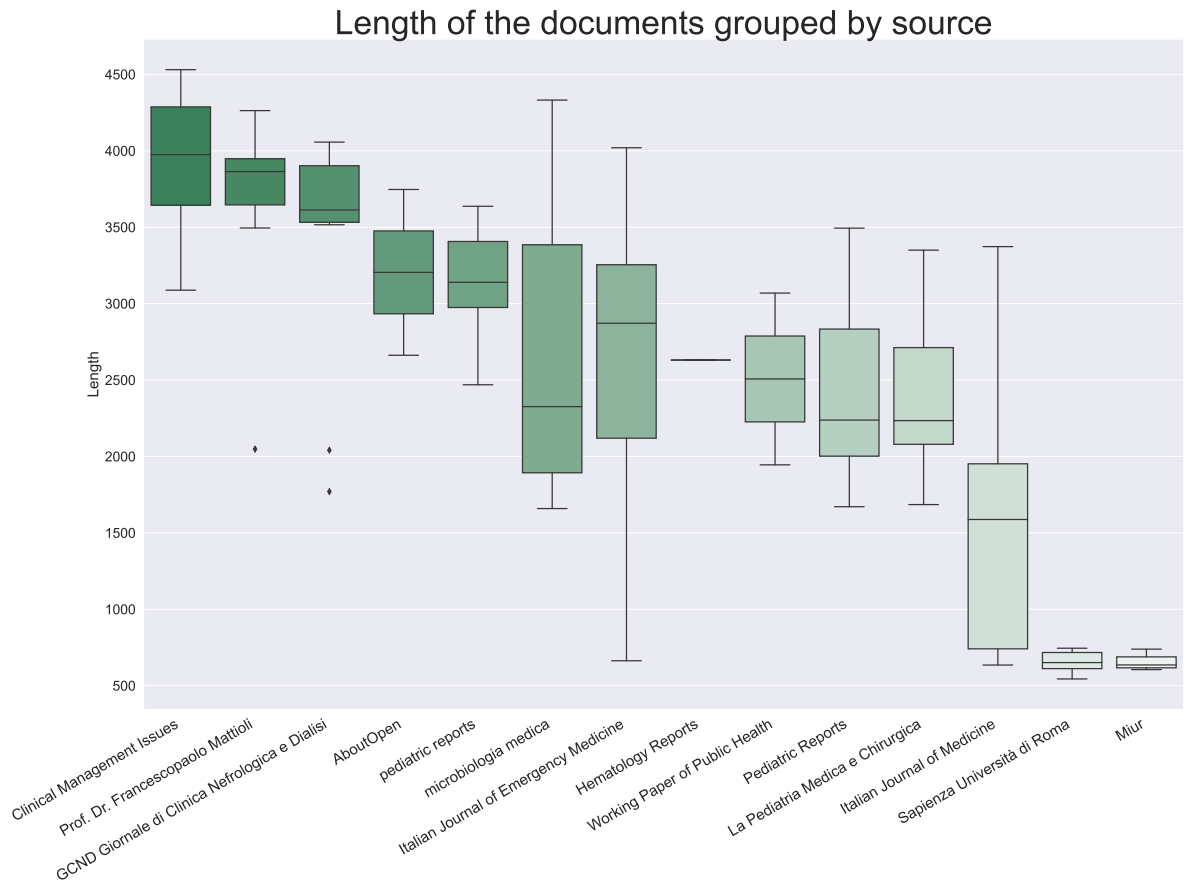


Figure 3.2: Boxplot showing the length of the documents grouped by source for layer 2. It is evident that "Miur" and "Sapienza Università di Roma" are the sources that correspond to shorter documents.

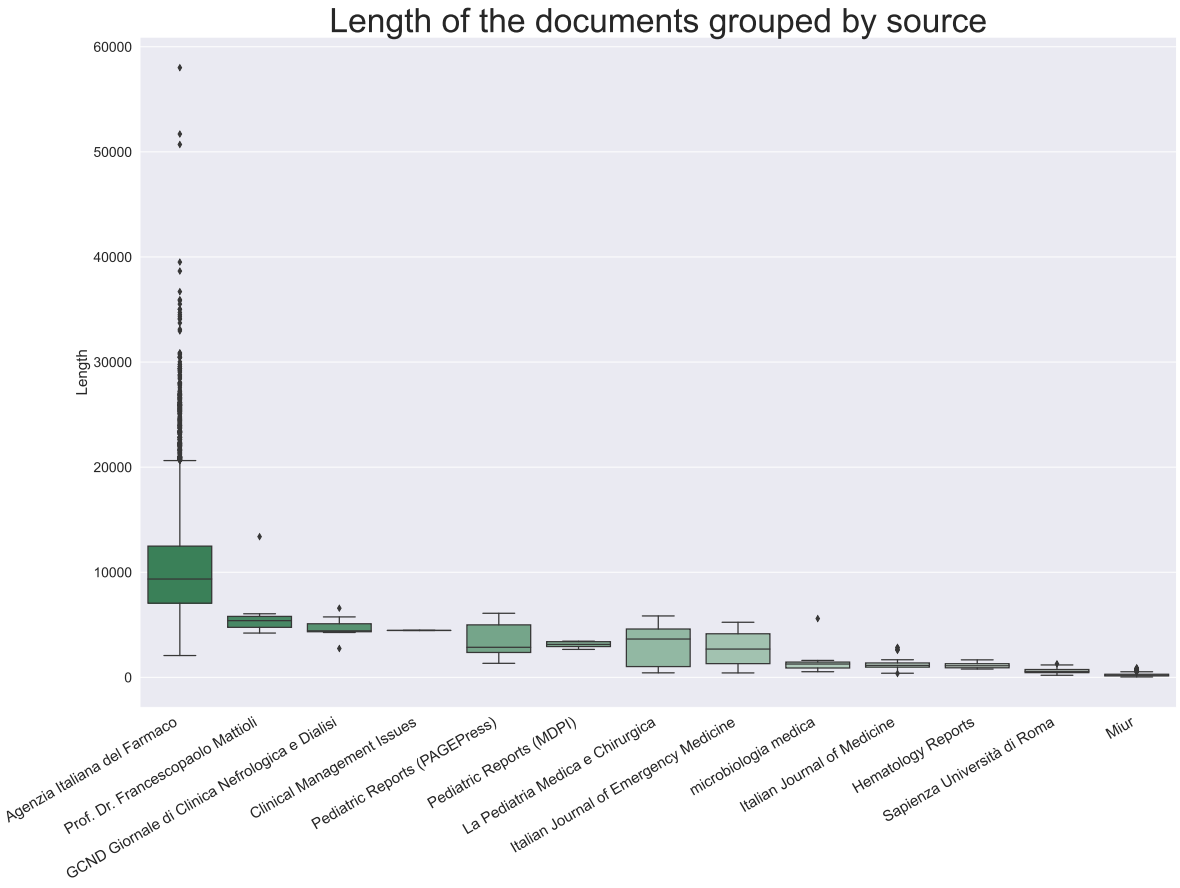


Figure 3.3: Boxplot showing the length of the documents grouped by source for layer 3. It is evident that "Miur" and "Sapienza Università di Roma" are the sources that correspond to shorter documents. AIFA documents are by far the longest documents.

4 | Methodologies

In this chapter, we will present the methodologies adopted to cluster E3C medical documents. The steps to obtain the results follow a sequential approach, which is summarized in Figure 4.1. Starting from the raw text data, the information is processed to get rid of the noisy components of the text, the words are translated in a mathematical framework and the texts are represented in a graph format. The preprocessed data texts are then fed to the InfoGraph model which creates a dense representation of each text. The document embeddings are then clustered by means of classical clustering techniques. Lastly, non-ordinary techniques are used to evaluate the goodness of the cluster assignment.

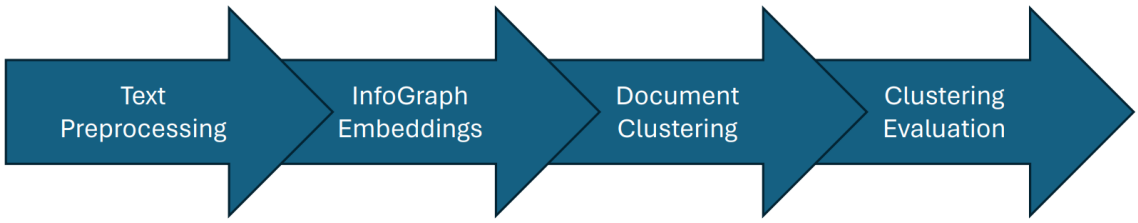


Figure 4.1: Methodology pipeline.

4.1. Data Preprocessing

In this section, we will present the methodologies that are needed to transform the E3C dataset in such a way that is possible to apply GNNs to it. Indeed the E3C dataset is a corpus of text documents and therefore is impossible to directly apply any mathematical model to it, but instead, a conversion from a *textual* to a *mathematical* representation of the documents is needed. To apply GNNs is also mandatory to have inputs that are in graph form, for this reason, we will build a graph for each document. In addition to those

necessary motivations, data preprocessing is also useful to obtain the best result out of the data that we are given, minimizing noisy information by removing errors, outliers, or nonimportant components of the input data. All those aforementioned procedures will be explained in the following subsections. In Figure 4.2 is represented a schema of the data preprocessing procedure. In the following sections, will be described each step of the methodology pipeline.

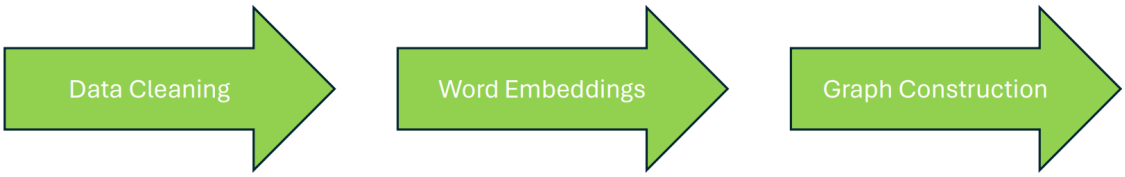


Figure 4.2: Steps of the data preprocessing procedure.

4.1.1. Data cleaning

As described in the previous chapter, we will consider from now on only the Italian subcorpus of E3C. For this part, layer 1, layer 2, and layer 3 will be considered in the same way, since they share the goal of cleaning each text from errors, not important words and punctuation. We have to proceed with this preprocessing procedure for the following reasons:

1. Removing less relevant words, leads to a better efficiency of the model, for example, articles and conjunctions appear very often in any kind of sentence but they don't add much to the meaning. On the other hand, words that are too rare, also need to be removed since the models don't have enough information to learn a good representation;

2. With fewer words, the computational demand of the algorithms decreases, leading to faster results.

Given a text document, the first operation to do in order to proceed with NLP models is *tokenization*. With this procedure, the text is split into smaller components called *tokens*, they could be sentences, words, or characters. We used word tokenization implementing the code with NLTK python library [5][1].

[1] `https://www.nltk.org/`

a	degl	lui	qua	sui
ad	degli	lungo	quale	sul
adesso	dei	ma	quanta	sul
agl	del	ma	quante	sull
agli	dell	me	quanti	sulla
ai	della	meglio	quanto	sulla
al	delle	mi	quarto	sulle
all	dello	mia	quasi	sullo
alla	dentro	mie	quattro	suo
alle	deve	miei	quella	suoi
allo	devo	mio	quelle	tanto
allora	di	molta	quelli	te

Figure 4.3: Subset of the Italian Stopwords.

The next step that we adopted in order to refine the tokenization is to remove the words that are not meaningful for the analysis. To clarify this better, consider the simple sentence *Today was a lovely day*, the word "a" doesn't contribute much to the meaning of the whole sentence so is better to remove it. Doing so for all the sentences, we will have a smaller number of tokens but of better quality. The words considered useless in this work, the so-called *Stopwords* were taken from NLTK's Italian Stopwords list and then removed. They include articles, conjunctions, etc..., see Figure 4.3 for a subset of the list of Italian Stopwords.

The last procedure that we adopted to refine tokens is *Stemming*. Its goal is to assign to similar words, or words with the same root, the same token. For example, *Running*, *Ran*, *Run* will be reduced to the same token *Run*; *Lucky*, *Luckly*, *Luck* will be reduced to the same token *Luck*. As we can see, the meanings of the words are slightly different, but the high-level concept is the same. Without this precaution, similar words will be considered in a totally distinct way, which is probably not the best; the fundamental reason to do Stemming is simplification: reducing the noise of the input data will increase the performance of the model. To perform Stemming, Snowball Stemmer from NLTK was used [52][2]. We observe that Stemming is just one way to map different words to the same *stem*, there are other techniques such as *Lemmization* that can be used with a similar scope. At the end of those steps, the corpus is tokenized, filtered of stopwords, and each word has a high-level meaning associated with it. In the next section, we will show how

[2] `https://snowballstem.org/`

to *translate* each word to a mathematical framework, using *Word Embeddings*.

4.1.2. Word Embeddings

Word embeddings are vector representations of words in a continuous vector space. In the last decade, many architectures have been developed to obtain word embeddings since they proved to be useful for a rich variety of tasks [45] [51]:

- Word embeddings are commonly utilized as a crucial step in training NLP models, primarily due to the nature of deep learning algorithms, which require numerical data as input. By converting words into numerical vectors, word embeddings capture semantic similarities and contextual information, enabling the model to better understand the meaning of words within a given context;
- Having a mathematical representation of words allows to treat words in the same way of numbers, so any operation is possible. As an example the expected result of word embedding is to have that "king" – "man" + "woman" = "queen", this is verified in almost every model, given that a corpus big enough is provided;
- Words can be represented graphically after reducing the dimension of the embedding space using for example t-SNE (t-distributed stochastic neighbor embedding) or PCA (principal component analysis). These visualizations can provide insights into semantic relationships and clusters among words;
- Prove similarity between words: any kind of similarity measure that can be applied to vectors can be applied to words (e.g. cosine similarity);
- Word and text translations can also be performed by means of word embeddings, given two different corpora for each language ([10]).

One of the first and most popular word embedding is Word2Vec by Mikolov et al. [45]. They proposed two similar architectures: CBOW and Skip-Gram. In short, in CBOW (Continuous Bag Of Words), each word is predicted using the neighboring words, in Skip-Gram surrounding words are predicted given the central word. The ideas are the opposite but the results are similar, leading to word embeddings for each word.

After that, other methods were developed, for example, GloVe (Global Vectors for word representation) [51] and BERT (Bidirectional Encoder Representation from Transformers) [12], confirming the importance of word embeddings in many scenarios.

For this work, we stuck to Word2Vec over BERT because:

- the standard dimension of BERT embeddings is 768, and this value cannot be easily

modified because it comes from BERT architecture itself, while Word2Vec allows to choose the size of the embeddings, in the direction of a smaller size to reduce the computational load for models.

- The main characteristic of BERT embeddings is that different contexts in which the same word appears in the text lead to different word embeddings for the same word. For this reason, BERT embeddings are identified as *dynamic word embeddings*, and Word2Vec embeddings as *static word embeddings*. By changing the embeddings for the same word which appears in different positions, there is not anymore the possibility to use a single node in the graph for every distinct word in the text, but rather a new node should be created for every word in the text.

- Considering the dynamic nature of BERT embeddings, for the same word appearing in different positions in the text and in different documents, there is the need to recompute every time the embedding and is not possible to reuse the same embedding as it happens for static word embedding methods. This leads to a massive overconsumption of computational power, which is also increased considering the high complexity of the BERT model in itself.

- Additionally, the work by Bugueno et al. [8], showed that in a setting with graphs as data, pre-trained static word embeddings, instead of BERT vectors, allow reaching outstanding results on some tasks, this finding strengthens the choice of our simple yet effective Word2Vec model for word embeddings.

In particular, we choose the CBOW model from Word2Vec which is a feedforward neural network architecture that aims to predict a target word given its context words. See Figure 4.4 for a visual representation of the architecture. The basic architecture of CBOW can be summarized in the following points:

- Input Layer: The input layer represents the context words. Each input is represented as a one-hot encoding vector, i.e. the vector has the size of the vocabulary and all the values are 0s except for the position which identifies the word in analysis, which is represented with a 1. The number of input neurons is determined by the size of the context window, which specifies how many words should be considered to the left and right of the target word.

- Projection Layer: In the projection layer, each input word is converted into a dense, fixed-size vector representation. These word vectors are the *word embeddings* and are learned during the training process.

- Hidden Layer: the number of neurons in the hidden layer is not fixed, its purpose is

to capture the relationships between the context words and the target word. This layer performs a linear transformation on the word embeddings.

- Output Layer: The output layer consists of as many neurons as there are unique words in the vocabulary. Each output neuron corresponds to a specific word in the vocabulary, and the *softmax* activation function is applied to produce a probability distribution over the vocabulary. The word with the highest probability is the predicted target word based on the surrounding context words.

One weakness of CBOW is that it loses all information about word order: *John likes Mary* and *Mary likes John* correspond to identical vectors. Second, the model does not attempt to learn the meaning of the underlying words, and as a consequence, the distance between vectors doesn't always reflect the difference in meaning. An example is: *He died due to a heart attack* and *He memorized all the book by heart*; in this context, the word *heart* takes two completely different meanings that a static model like Word2VecCBOW cannot capture while a dynamic model like BERT can capture, as previously explained. Nevertheless, CBOW performs surprisingly well and so we decided to use this simple version, keeping in mind those two drawbacks. In Table 4.1 and in Table 4.2, are shown some examples of the similarity between words taken from the E3C dataset. Generally, the results show how well CBOW embeddings are capable of capturing similarity between concepts.

Ita Target	**Eng Target**	**Ita Context**	**Eng Context**	**Similarity**
cancro	cancer	cmv	cmv	0.942
cancro	cancer	positivo	positive	0.941
cancro	cancer	tiroide	thyroid	0.934
cancro	cancer	mieloma	myeloma	0.931
cancro	cancer	ovarico	ovarian	0.929
cancro	cancer	leucemia	leukaemia	0.929
cancro	cancer	linfoma	linfoma	0.928
analgesico	painkiller	tachipirinaflu	tachipirinaflu	0.990
analgesico	painkiller	esitol	exitol	0.989
analgesico	painkiller	enturen	enturen	0.984
analgesico	painkiller	bor	bor	0.984
analgesico	painkiller	neodidro	neodhydro	0.983
analgesico	painkiller	Mediwound	Mediwound	0.981
analgesico	painkiller	cerebellare	cerebellar	0.979

Table 4.1: Top seven most similar context words related to the target word according to cosine similarity of the CBOW embeddings. The target words in this example are "Cancer" and "Painkiller".

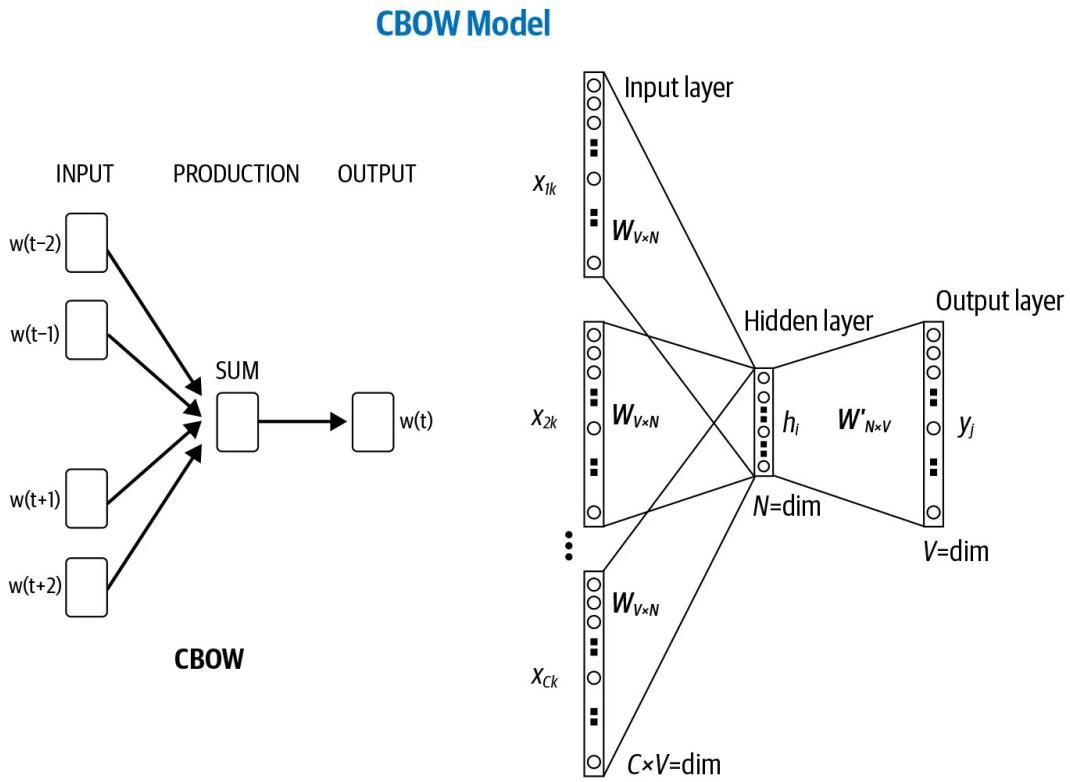


Figure 4.4: CBOW architecture. On the right, there is a focus on the dimensions of each component. Picture taken from `https://medium.com/the-modern-scientist/understanding-the-continuous-bag-of-words-cbow-model-586c5f60cb0d`.

From the implementation point of view, CBOW's embeddings were produced from the tokenized corpus of E3C, as explained in 4.1.1, with GENSIM python library [56][3] using a window size of 8 and producing vector representations of dimension 10.

4.1.3. Graph construction

Considering the advantages of graph structure to represent textual documents and its success in the field of supervised classification, as is explained deeply in Section 2.6, for this thesis work, where the goal is to produce unsupervised clustering, graph-based models are used to represent E3C documents.

The work by Bugueno et al. [8], provides an objective comparison of the different graph-based representations of text, explaining for each application which graph structure works better, their results were important for our structure choice. They have shown that "intuitive graphs" (i.e. graphs that are based solely on simple relationships between pairs

[3] `https://radimrehurek.com/gensim/`

Ita Target	Eng Target	Ita Context	Eng Context	Similarity
cancro	cancer	capelli	hair	0.374
cancro	cancer	diagnosi	diagnosis	0.641
cancro	cancer	medicinale	medicine	-0.424
cancro	cancer	infermiere	nurse	-0.051
cancro	cancer	febbre	fever	0.268
analgesico	painkiller	capelli	hair	0.146
analgesico	painkiller	diagnosi	diagnosis	-0.300
analgesico	painkiller	medicinale	medicine	0.602
analgesico	painkiller	infermiere	nurse	0.183
analgesico	painkiller	febbre	fever	0.365

Table 4.2: Similarity according to cosine similarity of the CBOW embeddings between "Cancer" and "Painkiller" target words related to five random medical words.

of nodes and only consider basic co-occurrence statistics if needed) like the one from TextLevelGCN, reach results across all considered tasks, especially for longer documents, exceeding those of BERT and other transformer-based architectures, while the performances obtained with more complex graphs are similar.

The E3C dataset contains generally long documents with few exceptions, so following, the previous results, an intuitive graph seems to be a good choice. Moreover, considering that the E3C dataset contains thousands of documents, we choose a graph representation that is not overly complex, in order to have faster results, and less memory consumption while still preserving global information. For the aforementioned motivations, we adopted a graph structure similar to TextLevelGCN [78], where each word represents a node in the graph, each edge represents the link between neighboring words and each graph represents a document.

In the following paragraphs, we will explain with more details our graph structure:

We denote a text with l words as $T = \{r_1, ...r_i, ..., r_l\}$, where r_i denotes the representation of the i_{th} word. r_i is a vector initialized with d dimension word embedding see Section 4.1.2 and can be updated by training; r_i. To build a graph for a given text, we regard all the words that appeared in the text as the nodes of the graph. Each edge starts from a word in the text and ends with its adjacent words. Concretely, the graph of text T is defined as:

$$N = \{r_i | i \in [1, l]\} \tag{4.1}$$

$$E = \{e_{ij} | i \in [1, l]; j \in [i - p, i + p]\} \tag{4.2}$$

where N and E are the node set and edge set of the graph. Here p denotes the number of adjacent words connected to each word in the graph. Is important to notice that we explicitly wrote p even if we will use $p = 1$, to present definitions in the same way of TextLevelGCN, and to allow future works to generalize our model to any chosen p. We observe that TextLevelGCN authors, explained in their paper that the graph classification accuracy is slightly increasing, by increasing the parameter p, until it reaches the value $p = 3$, to then drop rapidly. We consider also that this parameter is used to enhance local context learning within neighboring words and that firstly we used CBOW embeddings which in an analogously way capture information of neighboring words that are inside a certain window size and secondly having multiple graph convolutional layers that act as message passing within nodes, we would expect for those reasons to not notice big changes in performances by chancing p, having enough local context knowledge in proximate words.

The adjacency matrix A is defined in the following way:

$$a_{ij} = \begin{cases} 1 & \text{if } e_{ij} \in E \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

On the other hand, TextLevelGCN doesn't build an adjacency matrix to store information on the edge relation between words; instead, the authors created a global shared matrix of weights, in the sense that each edge is associated with a number between 0 and 1. This edge weight matrix is learned during training, so its values change, unlike our "classical" adjacency matrix.

In Figure 4.5 an example of graph structure taken from TextLevelGCN [26].

With the choice of $p = 1$, the graph structure ends up following "Random-Walk Term Weighting for Improved Text Classification" [22] presented by Hassan and Banea. An example of graph structure is presented in Figure 4.6. All the documents from the E3C dataset will be transformed in the same way of this example, where each node represents words, an edge connects adjacent words, and when a new word is encountered, a new node is added to the graph.

Comparing TextLevelGCN to other graph representations, it has the benefit of consuming less memory by connecting words in a small contextual window, because it excludes a good many words that are far away in the text and have little relation with the current word and thus significantly reduces the number of edges. The message-passing mechanism makes nodes in the graph perceive information around them to get precise meaning in a specific context.

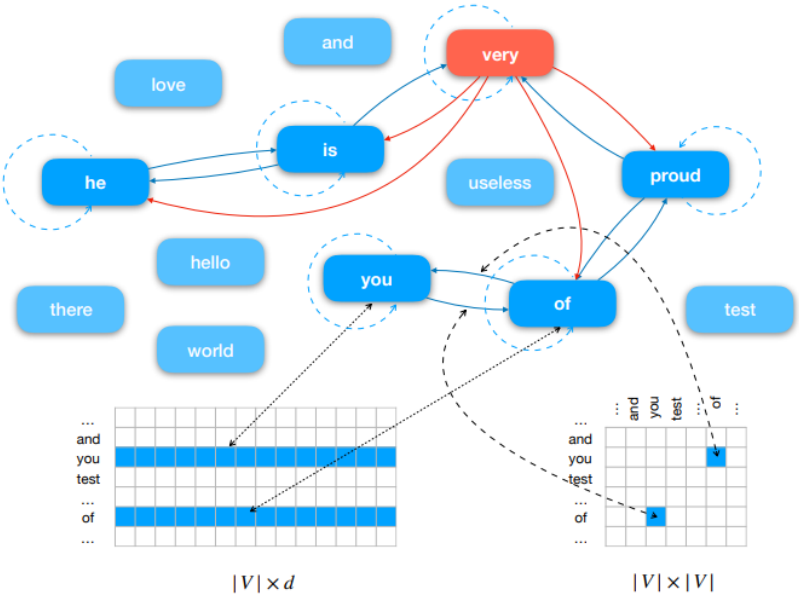


Figure 4.5: Structure of graph for a single text "he is very proud of you.". In the bottom left is depicted the feature matrix containing the word embeddings for each word, and in the bottom right is depicted the adjacency matrix showing the edges between each word in the text. For the convenience of the display, in this figure, we set p = 2 for the node "very" (nodes and edges are colored in red) and p = 1 for the other nodes (colored in blue). In actual situations, the value of p during a session is unique. Picture from TextLevelGCN [26].

From the implementation point of view, given the documents of the E3C dataset, the adjacency matrix of each graph, was created in a way that each node represented a specific, unique, word, in order to avoid multiple nodes corresponding to the same word.

With the use of *torch geometric* library [17][4], a *Data* object was created for each text containing the information of the adjacency matrix and of the feature matrix. After creating a list containing all Data objects of the corpus, the *DataLoader* method was used, creating a graph dataset ready to be deployed for the next Graph Neural Network models that we will present in the next Section 4.2.

[4] `https://github.com/rusty1s/pytorch_geometric`

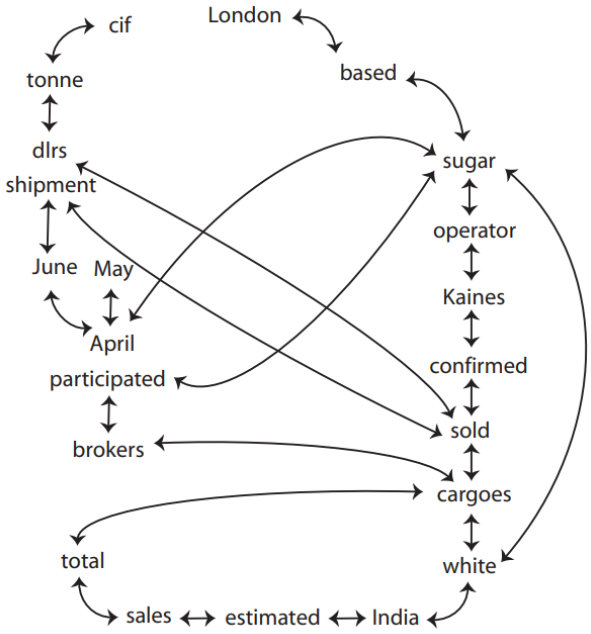


Figure 4.6: Structure of graph for the sample text *"London-based sugar operator Kaines Ltd confirmed it sold two cargoes of white sugar to India out of an estimated overall sales total of four or five cargoes in which other brokers participated. The sugar, for April/May and April/June shipment, was sold at between 214 and 218 dlrs a tonne cif, it said."*. Picture taken from Hassan and Banea paper [22].

4.2. Infograph

In this section, we will present the model that we used to cluster documents from the E3C corpus: InfoGraph.

InfoGraph is a self-supervised model, created by Sun et al. [64]. Its goal is to learn the whole representation of graphs in both unsupervised and semi-supervised settings. We remind that semi-supervised learning is the learning schema where very few labeled data are given and the model learns from both the labeled and the unlabeled data. Since in the E3C corpus no labels at all are provided, we will only consider the unsupervised version of InfoGraph.

InfoGraph model takes inspiration from the contrastive self-supervised learning model called Deep Graph Infomax [70], where the aim is to find node embeddings. Since the authors of InfoGraph wanted to focus on graph-level embeddings, they adopted a similar model to Deep Graph Infomax that is able to address the different task. To differentiate the two models the authors choose the name of InfoGraph.

4.2.1. Motivations on the choice of InfoGraph

InfoGraph is not a model that is tailored for natural language processing tasks, indeed one of its purposes is to be able to work across different graph datasets belonging to different fields, with different numbers of nodes and graphs. For this reason, the authors tested the model on chemical molecules datasets (MUTAG, PTC), movie collaboration datasets (IMDB-MULTI, IMDB-BINARY), and online discussion threads (REDDIT-BINARY, REDDIT-MULTI5K), to prove the model performance across different fields.

We note that the authors of InfoGraph evaluated the embeddings over supervised classification as downstream task, but we highlight that InfoGraph is purely unsupervised, just the classification downstream task is not. Since E3C doesn't have labels, we don't evaluate the embeddings by means of a classification downstream task, but instead by a clustering algorithm. To the best of our knowledge, no research in literature has been done focusing on unsupervised downstream task with Infograph. Nevertheless, since InfoGraph has been proven to produce valuable embeddings on graph classification tasks, and considering that embeddings creation with InfoGraph and classification output produced with any classifier (e.g. Logistic Regression, Random Forest, MLP) are two totally separated learning phases, we expected InfoGraph embeddings to perform well also in unsupervised downstream tasks.

No testing was provided on the corpus dataset by the authors, neither other researches have been conducted by different authors regarding the use of InfoGraph for pure NLP tasks. In 2023 Seki et al. [61] deployed InfoGraph to transform heterogeneous non-fixed length clinical information of patients into fixed size embeddings in an unsupervised way, to then apply a multi-layer-perceptron to those embeddings and predict patient readmission in the hospital in a supervised fashion. This finding motivates once again the choice of InfoGraph since, even if the authors focused on heterogeneous clinical data of the patients such as medicines, admission date in the hospital and disease name, some of the information have been provided in a textual format, transforming words with word2vec model. In addition to this NLP motivation, the field of the analysis, i.e. clinical information of patients, is very close to the field of the E3C dataset. For the reasons mentioned above, this finding underscores the potential effectiveness of employing InfoGraph in the context of our analysis.

As explained in Section 2.4, many techniques that apply self-supervised learning on graphs have been developed, for example, GraphCL [20] and Attribute Mask [28], but InfoGraph was chosen over the others because of the possibility to apply a fully unsupervised approach to the data and because of its relatively fast training time.

4.2.2. Model description

The definition of the problem of learning unsupervised embeddings of whole graphs is the following:

Definition 1. *Given a set of graphs* $\mathbb{G} = \{G_1, G_2, ...\}$ *and a positive integer* δ *(the expected embedding size), our goal is to learn a* δ*-dimensional distributed representation of every graph* $G_i \in \mathbb{G}$*. We denote the number of nodes in* G_i *as* $|G_i|$*. We denote the matrix representation of all graphs as* $\Phi \in \mathbb{R}^{|\mathbb{G}| \times \delta}$*.*

InfoGraph adopts Graph Neural Networks (GNNs) to learn node embeddings for each graph. The node representation learned with neighborhood aggregation of the features will be referred from now on as *patch representation*. From the embeddings of the nodes within a graph, a whole graph embedding is produced by a READOUT function that acts as a summarization of the patch representation into a fixed size graph-level representation, to which we will refer as *global representation*.

Mathematically the k-th layer of a GNN can be described in its general form as:

$$h_v^{(k)} = COMBINE^{(K)}\Big(h_v^{(k-1)}, AGGREGATE^{(K)}\Big(\Big\{\Big(h_v^{(k-1)}, (h_u^{(k-1)}, e_{uv}\Big) : u \in \mathcal{N}(v)\Big\}\Big)\Big). \tag{4.4}$$

Where h_v is the patch representation of node v, e_{uv} the feature vector of the edge and $\mathcal{N}(v)$ is the neighboring set of nodes of the node v. $h_v^{(0)}$, is the initialized node feature vector.

The objective is to obtain graph representations by maximizing the *mutual information* between graph-level and patch-level representations. In this way, the graph representation can learn to encode aspects of the data that are shared across all substructures. Assume that we are given a set of training samples $\mathbf{G} := \{G_j \in \mathbb{G}\}_{j=1}^N$ with empirical probability distribution $\mathbb{P}$ on the input space. Let ϕ denote the set of parameters of a k-layer graph neural network. After the first k layers of the graph neural network, the input graph will be encoded into a set of patch representations $\{h_i^k\}_{i=1}^N$. Next, we summarize feature vectors at all depths of the graph neural network into a single feature vector that captures patch information at different scales centered at every node, by using concatenation, this allows the embedding vector to contain information of every GNN layer. So the final patch level and graph level representations are:

$$h_{\phi}^{i} = CONCAT(\{h_{i}^{k}\}_{i=1}^{N}). \tag{4.5}$$

$$H_{\phi}(G) = READOUT(\{h_{\phi}^{i}\}_{i=1}^{N}). \tag{4.6}$$

h_{ϕ}^{i} is the patch level summarized embedding for the node i and $H_{\phi}(G)$ is the global summarized embedding for the graph G, produced with a READOUT function.

Before explaining the loss function used to train the model, we note that this model is similar to Deep Graph Infomax, but since its authors focused on learning node embeddings rather than whole graph embeddings, some important differences are present. First of all, as we discussed in Section 2.2, GIN [76] is the best GNN architecture in terms of graph expressive power, so using GIN is preferred when dealing with whole graph tasks. For this reason InfoGraph adopts GIN as graph convolutional encoder; while Deep Graph uses GCN [30], which works better with node-level tasks. For the same motivation, following the results of GIN, the READOUT function is chosen to be the sum over the mean.

Considering the above motivations, Equation 4.6, can be written in a more detailed version as follows:

$$H_{\phi}(G) = \sum_{i=1}^{N} h_{\phi}^{i}. \tag{4.7}$$

A graphical description of the graph convolutional encoder architecture can be seen in Figure 4.7, taken from InfoGraph original paper [64].

The mutual information (MI) estimator on global/local (i.e. graph/node) pairs is defined maximizing the estimated MI over the given dataset $\mathbf{G} := \{G_j \in \mathbb{G}\}_{j=1}^{N}$:

$$\hat{\phi}, \hat{\psi} = \arg\max_{\phi,\psi} \sum_{G \in \mathbf{G}} \frac{1}{|G|} \sum_{u \in G} I_{\phi,\psi}(h_{\phi}^{i}; H_{\phi}(G)). \tag{4.8}$$

$I_{\phi,\psi}$ is the mutual information estimator modeled by the discriminator T_{ψ} and parameterized by a neural network with parameters ψ. The Jensen-Shannon MI estimator, following the formulation of Nowozin et al. [48] is used:

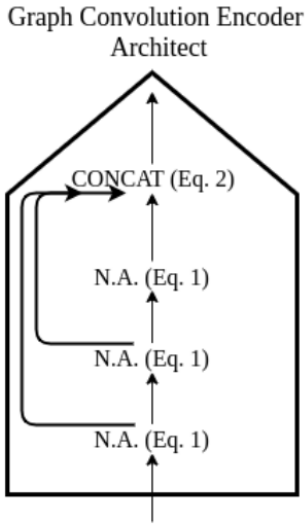


Figure 4.7: Visual description of the GNN encoder used in InfoGraph to create a patch level representation for each node. N.A. denotes neighborhood aggregation, the results obtained in each layer are then concatenated at the end of the encoder. Picture taken from InfoGraph original paper [64].

$$I_{\phi,\psi}(h^i_\phi(G); H_\phi(G)) := \mathbb{E}_{\mathbb{P}}[-sp(-T_{\phi,\psi}(h^i_\phi(x); H_\phi(x)))] - \mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}[sp(T_{\phi,\psi}(h^i_\phi(x'); H_\phi(x)))]. \tag{4.9}$$

Where x is an input sample, x' i.e. the negative sample is an input sampled from $\tilde{\mathbb{P}} = \mathbb{P}$, a distribution identical to the empirical probability distribution of the input space, $sp(z) = \log(1 + e^z)$ is the softplus function. In practice, negative samples are generated using all possible combinations of global and local patch representations across all graph instances in a batch.

Since the final graph embedding $H_\phi(G)$ is encouraged to have high MI with patches, i.e. local information, that contain information at all scales, i.e. global information, this favours encoding aspects of the data that are shared across patches and aspects that are shared across scales. By incorporating information from multiple scales and patches, the final graph embedding becomes a richer representation that considers both local and global information, this process leads to an high-level representation learning of the data and therefore a deeper understanding of the graphs.

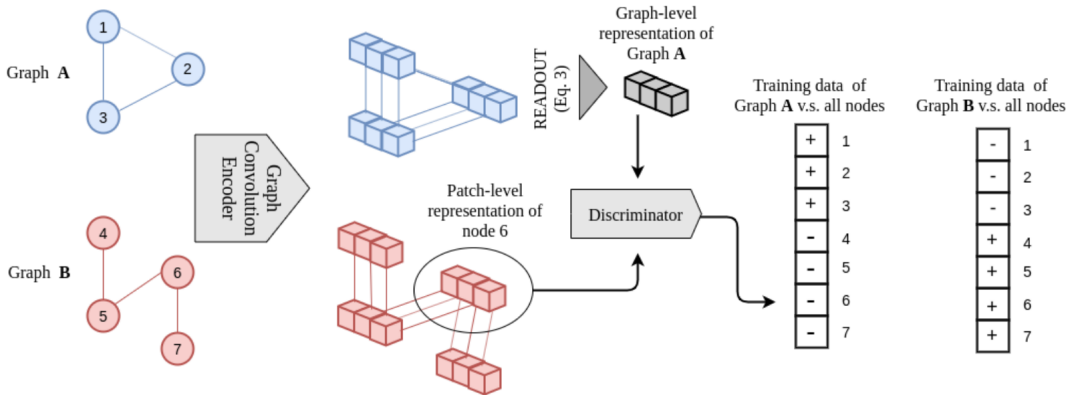


Figure 4.8: Description of the learning process of InfoGraph. Picture taken from InfoGraph original paper [64].

The algorithm is presented in Figure 4.8 using a toy example composed by graph A which has 3 nodes and graph B which has 4 nodes. Both graphs pass into the graph convolutional encoder presented in Figure 4.7, to learn patch representations of the nodes and global representations of the graphs. For each node of both graphs, a pair <patch representation, global representation> is given to the discriminator which tries to detect whether the node belongs to the graph or not, by using the labels in a self-supervised fashion. This example shows how negative and positive samples are created to then compute MI maximization by Equation 4.8. InfoGraph uses a batch-wise fashion to generate all possible positive and negative samples. Considering that only graph A and graph B belongs to the batch, there are 7 training input given to the discriminator for graph A and 7 training input given to the discriminator for graph B. Therefore the discriminator will take 14 pairs <patch representation, global representation> as input for this example. In real case scenarios, the procedure above explained has to be repeated for every pair of node and graph present in the batch leading to a much bigger training data for the discriminator.

4.2.3. InfoGraph for E3C

InfoGraph model, as explained in the previous sections, is able to produce a fixed-size vector of real number embeddings as output, given homogeneous graphs of different structures and dimensions. As we explained in Section 4.1.3, the input graphs fulfill those requirements and each graph describes one document of the E3C dataset. We worked in a transductive setting, where documents coming from both layer 1 and layer 3 were trained together by InfoGraph. In our implementation, we modified the code to remove the classification part and we collected the embeddings after the whole training phase to then feed a clustering layer with the aforementioned embedding.

The default hyperparameters of InfoGraph that we used for our analysis are Learning Rate = 0.001, Hidden Dimension = 16, Number of Graph Convolutional Layers = 3, and Number of epochs = 10. The chosen number of epochs is low because of early stopping, indeed, after a few epochs, the model stops decreasing the loss function.

We observe that the Hidden Dimension is the dimensionality of the embedding to which each node is transformed with a Graph Convolutional Layer. Since the summarization function chosen by GIN and InfoGraph (Equation 4.5, Equation 4.7) concatenates the embedding vectors of the nodes at every graph convolutional layer, the final dimension of the produced graph embeddings is: DIM(Graph Embeddings) = Number of Graph Convolutional Layers × Hidden Dimension. The motivation for this concatenation process is given by GIN's authors, who affirm that an important aspect of the graph-level readout is that node representations, corresponding to subtree structures, get more refined and global as the number of iterations increases. A sufficient number of iterations is key to achieving good discriminative power. Yet, features from earlier iterations may sometimes generalize better. To consider all structural information, graph representations concatenated across all layers of GIN are used.

Considering our specific case the produced graph embeddings have a dimension of DIM(Graph Embeddings) = 3× 32 = 96. The curse of dimensionality is the problem for which, having the data in a high dimensional space, leads to a sparsity of the data points, where the distance between them is exponentially increasing by increasing the number of features. In the next sections, we will discuss whether the curse of dimensionality may be an issue, considering the high dimensionality space in which our graph embeddings are.

4.3. Clustering

The produced document embeddings by InfoGraph, need now to be evaluated for the main purpose of this thesis work: document clustering.

We proceeded explaining the employed methodologies in order to perform the clustering of the 10213 documents of layer 3 and 86 documents of layer 1.

Since documents of layer 1 have clinical annotations and layer 3 doesn't, we will often consider layer 3 as a sort of training dataset and layer 1 as a test set. With the high numerosity of documents in layer 3 we can experiment different clustering techniques and then verify the hypothesis on layer 1 only.

In the whole analysis, we conducted t-distributed Stochastic Neighbor Embedding (t-SNE) by Van der Maaten et al. [67] for the visualization of the document embeddings

and their clusters. Other techniques such as Principal Component Analysis (PCA) have been tested but since InfoGraph is able to capture nonlinear relations within the data, and a linear model like PCA cannot capture all the information, other techniques are preferred in order to not lose information; furthermore, many principal components were needed to explain enough variability of the document embeddings but providing a visual representation in 2D or 3D would again lead to an information loss of the data.

Two different clustering approaches have been used: k-means [40] and hierarchical agglomerative clustering. For k-means, the choice of k has been made following the Silhouette score [57], the similarity measure used for k-means is *euclidean distance*. For hierarchical agglomerative clustering instead, the *cosine similarity* has been chosen to measure the similarity between the document embeddings. Cosine similarity has several advantages when applied to text data:

- Scale Invariance: Cosine similarity is scale-invariant, meaning it's not affected by the magnitude of the vectors. This makes it suitable for documents of different lengths.
- Angle Measure: It focuses on the direction of vectors rather than their absolute values, which is crucial for text similarity, where document length can vary.
- Efficiency: Calculating cosine similarity is computationally efficient, making it suitable for large-scale text datasets.

Euclidean distance was chosen for k-means for implementation issues with cosine similarity, since the algorithm is designed for this type of similarity measure.

As previously mentioned, we used the Silhouette score to find the proper number of clusters for k-means. Silhouette score is a method that measures how similar are the points in the same cluster relatively to how dissimilar are the points in the same cluster and the points in the other clusters. Specifically, considering having the data points divided into k clusters, for each $i \in C_I$, where C_I is the cluster I, let

$$a(i) = \frac{1}{|C_I| - 1} \sum_{j \in C_I, i \neq j} d(i, j). \tag{4.10}$$

be the mean distance between i and all other data points in the same cluster, where $|C_I|$ is the number of points belonging to cluster C_I, and $d(i, j)$ is the distance between data points i and j in the cluster C_I. We can interpret $a(i)$ as a measure of how well i is assigned to its cluster (the smaller the value, the better the assignment). We then define the mean dissimilarity of point i to some cluster C_J as the mean of the distance from i to

all points in C_J where $C_J \neq C_I$. For each data point $i \in C_I$, we now define

$$b(i) = \min_{J \neq I} \frac{1}{|C_J|} \sum_{j \in C_J} d(i,j). \tag{4.11}$$

to be the smallest mean distance of i to all points in any other cluster (i.e., in any cluster of which i is not a member). The cluster with this smallest mean dissimilarity is said to be the "neighboring cluster" of i because it is the next best-fit cluster for point i.

We now define a silhouette (value) of one data point i

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, if |C_I| > 1. \tag{4.12}$$

For $s(i)$ to be close to 1 we require $a(i) \ll b(i)$. As $a(i)$ is a measure of how dissimilar i is to its own cluster, a small value means it is well matched. Furthermore, a large $b(i)$ implies that i is badly matched to its neighboring cluster. Thus an $s(i)$ close to 1 means that the data is appropriately clustered. If s(i) is close to -1, then by the same logic we see that i would be more appropriate if it was clustered in its neighboring cluster. An $s(i)$ near zero means that the datum is on the border of two natural clusters.

The mean $s(i)$ over all points of a cluster is a measure of how tightly grouped all the points in the cluster are. Thus the mean $s(i)$ over all data of the entire dataset is a measure of how appropriately the data have been clustered.

4.4. Clusters Evaluation

Since no groundtruth labels are at the disposal of the E3C corpus, we had to find a way to evaluate the cluster assignment using non-ordinary methods. Classical methods for understanding the goodness of the clustering results such as *Within Sum of Squares (WSS)* or *Silhouette score* are not enough to evaluate the clusters. For instance, consider the scenario where two documents have different medical meanings but the produced embeddings of the two documents are similar. The two documents are assigned to the same cluster; using Silouhette score or WSS, it wouldn't be detected any error, but in reality the results are not correct. This happens because is not possible to rely on the accuracy of the word embeddings to evaluate the clustering results. In the next subsections, we will present the methods to pursue the goal, and in Chapter 5 we will present the results following those approaches.

4.4.1. UMLS similarity

As described in Section 3.3, the clinical entities of the documents belonging to layer 1 and layer 2 have been annotated according to UMLS. In this subsection, we show how the information about clinical entities has been exploited to evaluate the goodness of InfoGraph embeddings and their obtained clusters.

We recall that the goal of this thesis work is to create clusters of documents containing similar medical information. The clinical entities are tagged with CUI (Concept Unique Identifier) and as already explained, the documents haven't been provided with labels indicating the medical domain of the clinical case. With those two observations in mind, our idea was to make up for the missing labels of the documents by exploiting the information carried by the CUI. Indeed, some tools -which will be explained in the next paragraphs-, allow to measure the similarity between CUIs, and for example, by averaging the similarity between CUIs belonging to different documents, we can approximate the global similarity (in terms of medical meaning) between two documents. In this manner, we would not have labels for the documents describing their medical domain but at least a measure to compare similarities between pairs of documents. The outcomes can be used to evaluate how much InfoGraph was able to understand the similarity between documents (of layer 1 and layer 2).

To exploit CUI information and compute the similarity between clinical entities, we tried two approaches: an already implemented method, to which we will refer as *PyUMLSSimilarity* and a newly developed method which uses UMLS API (Application Programming Interface).

We used PyUMLSSimilarity[5] which is a package that computes a variety of semantic similarity metrics between concepts present in the UMLS database. It serves as a Python wrapper based on the Perl modules (UMLS Interface and UMLS Similarity) developed by Dr. Bridget McInnes and Dr. Ted Pedersen [44], offering an accessible and user-friendly interface for Python users.

PyUMLSSimilarity package allows to compute similarity between CUI in a rich variety of ways, an explanation of the possible methods is presented by Alonso et al. [1]. UMLS Metathesaurus is a very large, multi-purpose, and multi-lingual vocabulary database that contains information about biomedical and health-related concepts, their various names, and the relationships among them. In other words, it is a vocabulary containing medical terms and their relations.

[5] `https://github.com/victormurcia/PyUMLS_Similarity?tab=readme-ov-file`

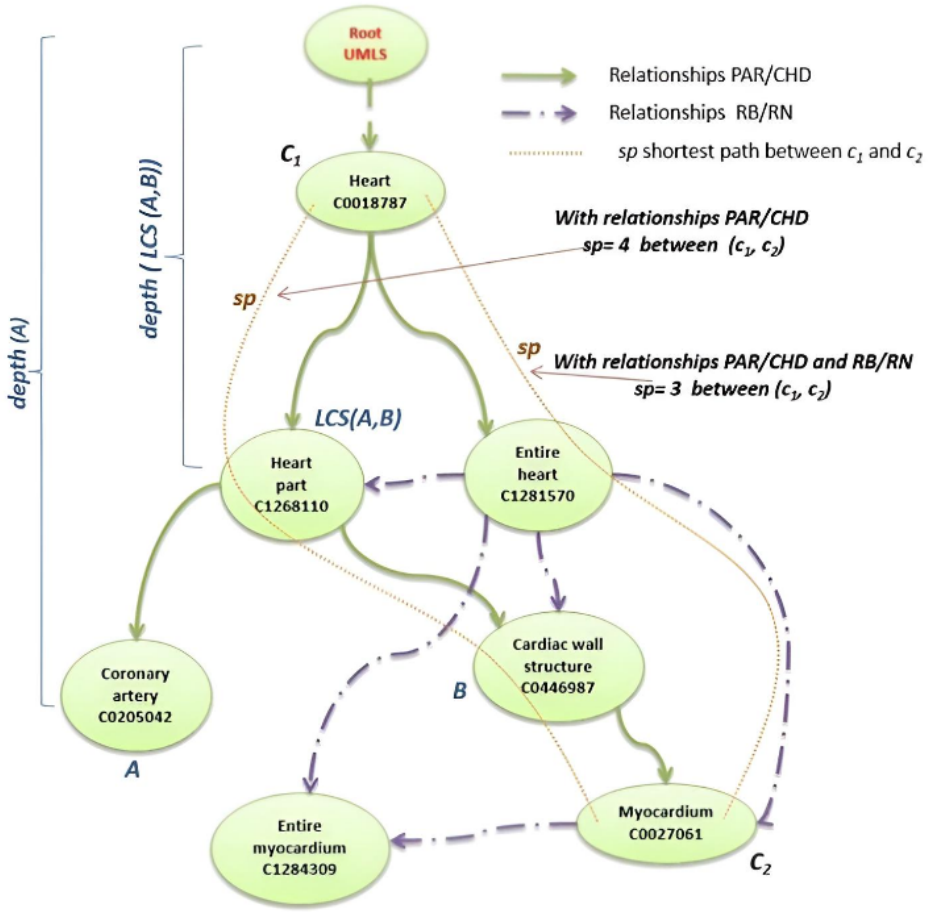


Figure 4.9: Example of hierarchical relationships between concepts in UMLS Metathesaurus.

UMLS Metathesaurus is in the form of a net, in the sense that every medical term belonging to UMLS Metathesaurus is linked to every other term following the path composed by the edges between terms. The edges represent the different types of relations between the terms. Hierarchical relationships cover either direct synonymical relations (Parent/Child (PAR/CHD) type) or indirect ones (Broader/Narrower (RB/RN) type). Following different types of relations the path between two concepts can vary. An example of how UMLS Metathesaurus and the relations between its concepts appear is presented in Figure 4.9. As represented, the shortest path (sp) from c_1 to c_2 using PAR/CHD relations has length 4, while using both PAR/CHD and RB/RN the path has length 3. LCS refers to Least Common Summoner also known as Least Common Ancestor (ACA).

From the UMLS Metathesaurus, many similarity metrics can be used to evaluate the similarity between two concepts. The first work to find a metric in semantic nets (such as UMLS Metathesaurus) is from Rada et al. [54], which define the metric as the number of nodes in the shortest path between two concepts, as shown in Formula 4.13.

$$SIM(c_1, c_2) = sp(c_1, c_2). \tag{4.13}$$

Pedersen et al. [50] developed the *Path Measure* which modifies the previous equation to have normalized values for the similarity between concepts $\in [0, 1]$, as shown in Formula 4.14.

$$SIM(c_1, c_2) = 1/sp(c_1, c_2). \tag{4.14}$$

Many other techniques have been developed to evaluate the similarity between concepts, for example, Leacock et al. [34] developed a metric that scales logarithmically to the total depth of the taxonomy. Thus, the deeper the taxonomy (that is, the more complex and thorough), the larger the relative value of semantic proximity between two terms would be.

Out of the possible metrics to evaluate the similarity between CUI concepts corresponding to the clinical entities present in the documents of layer 1 and layer 2, Path Measure (Formula 4.14) has been chosen because it provided faster results.

Formally, to evaluate the similarity between two documents doc_a and doc_b belonging to layer 1 or layer 2, all the clinical entities have been extracted from the two documents, to then create two separated lists of CUIs, one for each document. For each CUI in the first list, the Path Measure has been measured with all the other CUIs from the second list and averaged. The results for each CUI in the first list represent the similarity between the CUIs of the first and second documents. Averaging the results, a measure of similarity between doc_a and doc_b is obtained. In Formula 4.15, is presented the mathematical representation of the previous explanation. The notation $|doc_a|$ is referring to the number of CUIs in doc_a. Computing the similarity of the pairs of documents allows to create the similarity matrix of the documents. Comparing the cosine similarity matrix created from the InfoGraph embeddings with the one created by PyUMLSSimilarity, we have a way to evaluate InfoGraph model based on the clinical entities belonging to the documents.

$$docSIM(doc_a, doc_b) = \frac{1}{|doc_a|} \sum_{CUI_i \in doc_a} \frac{1}{|doc_b|} \sum_{CUI_j \in doc_b} SIM(CUI_i, CUI_j). \tag{4.15}$$

The results for the E3C dataset, following this method, couldn't be evaluated in this thesis work because of the computational time requested for handling the operations done by the PyUMLSSimilarity package, indeed to find the Path Measure similarity (i.e. the

fastest similarity measure) between two CUIs more than 100 seconds were required, and considering that each document has approximately 10 CUIs in mean, using Formula 4.15, to find the similarity between two documents the total amount of time needed is in the order of $10 \times 10 \times 100 = 10000 second \simeq 2.7 hour$. To have the full similarity matrix of size 86×86, the process of computing the similarity between two documents should be repeated 85×43 times (since the similarity with itself is 1 and the similarity matrix is symmetric). The final computation time to compute the similarity matrix according to PyUMLSSimilarity, using Path Measure, only for layer 1 is therefore approximately $2.7 \times 85 \times 43 = 9868.5 hour$. With those simple calculations regarding the time complexity of the method, it was possible to conclude that the procedure was unfeasible, at least with a normal laptop, and even with higher computational capabilities it would not be feasible, considering the multiple experiments to carry out. In the next paragraphs, we will explain how we decided to bypass the computational time issue while still using information about the clinical entities and UMLS.

The second UMLS method to evaluate InfoGraph Embeddings is UMLS API[6]. UMLS website[7] is indeed able to present some valuable information when a CUI is searched, for instance, its AUIs and its broader and narrower concepts. Employing UMLS API, our idea was to find a new way of computing similarity between CUIs: for each CUI in the documents, a list of broader concepts can be extracted, and if another document has CUIs with broader concepts *similar* to the one under study, we can conclude that the two documents are similar.

To explain thoroughly the above idea, consider again two documents doc_a and doc_b belonging to layer 1 or layer 2; both documents have a list of CUIs, and from them, the sets containing the broader concepts of each CUI for both documents are stored in memory thanks to UMLS API. The multiple sets of both documents are then algebraically united to form two sets containing all the broader concepts of all the CUIs belonging to the two documents, separately. Then, the *Jaccard Index* is used to evaluate the similarity between the two sets of broader concepts and therefore the similarity between the two documents. The Jaccard Index (see Formula 4.16) is used over the simple intersection because it allows having a normalized number $\in [0, 1]$ as a result, and because it is considering also the relative context having the union as a denominator so that sets with higher cardinality are not advantaged.

[6] `https://documentation.uts.nlm.nih.gov/rest/home.html`
[7] `https://uts.nlm.nih.gov/uts/umls/home`

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}. \tag{4.16}$$

From a measure of similarity between two documents is then possible to create a similarity matrix SIM_{UMLS} (see Formula 4.17) containing the information about the similarity of each pair of documents. In Subsection 5.1.2, the similarity matrix based on UMLS will be used to evaluate InfoGraph embeddings.

$$SIM_{UMLS}(i, j) = J(i, j). \tag{4.17}$$

4.4.2. Doc2Vec

While Graph Neural Networks proved to obtain state of the art results in graph classification, there are no attempts in the literature of GNNs applied to a fully unsupervised setting. For this reason is impossible to compare InfoGraph results with other unsupervised GNNs methods. Nevertheless, in the framework of unsupervised clustering of textual documents, many approaches which don't include representing the texts as graphs, have been produced during the last two decades. One example is *Doc2Vec* model [33], which is an extension of word2vec model, already presented in Subsection 4.1.2. While the word2vec model creates fixed size embeddings of single words within a text, the doc2vec model creates fixed size embeddings of the whole documents within a corpus, which is exactly the same aim of InfoGraph. Following Doc2Vec original paper we explain its structure and possible variants. We observe that the authors often refer to *paragraphs* instead of *documents* in order to explain that all types of sequences of words can be used (sentences, paragraphs and whole documents).

Doc2Vec approach for learning paragraph vectors (i.e. document embeddings) is inspired by Word2Vec's methods for learning word vectors (i.e. word embeddings). The inspiration is that the word vectors are asked to contribute to a prediction task about the next word in the sentence. So despite the fact that the word vectors are initialized randomly, they can eventually capture semantics as an indirect result of the prediction task. A similar idea is used to create document embeddings with Doc2Vec: the paragraph vectors are also asked to contribute to the prediction task of the next word given many contexts sampled from the paragraph. The paragraph vector represents the missing information from the current context and can act as a memory of the topic of the paragraph. In the Doc2Vec framework, every paragraph is mapped to a unique vector, and every word is also mapped to a unique vector. The paragraph vector and word vectors are averaged or concatenated

to predict the next word in a context.

The paragraph token can be thought of as another word. It acts as a memory that remembers what is missing from the current context – or the topic of the paragraph. For this reason, this model is called the Distributed Memory Model of Paragraph Vectors (PV-DM). The contexts are fixed-length and sampled from a sliding window over the paragraph. The paragraph vector is shared across all contexts generated from the same paragraph but not across paragraphs. The word vector matrix, however, is shared across paragraphs. I.e., the vector for *powerful* is the same for all paragraphs.

The paragraph vectors and word vectors are trained using stochastic gradient descent and the gradient is obtained via backpropagation. At every step of stochastic gradient descent, one can sample a fixed-length context from a random paragraph, compute the error gradient from the network and use the gradient to update the parameters in our model. At prediction time, one needs to perform an inference step to compute the paragraph vector for a new paragraph. This is also obtained by gradient descent. In this step, the parameters for the rest of the model, the word vectors and the softmax weights are fixed.

Suppose that there are N paragraphs in the corpus, M words in the vocabulary, and we want to learn paragraph vectors such that each paragraph is mapped to p dimensions and each word is mapped to q dimensions, then the model has the total of $N \times p + M \times q$ parameters (excluding the softmax parameters). Even though the number of parameters can be large when N is large, the updates during training are typically sparse and thus efficient.

After being trained, the paragraph vectors can be used as features representing the information hidden in the documents. Those features can then be used to train standard algorithms for text classification or text clustering [32].

We notice that Doc2Vec is a totally unsupervised algorithm so no labels are needed. The algorithm is also capable of learning word embeddings as well as document embeddings and the produced word embeddings generally perform better than simple word2vec since with Doc2Vec the information about the document is useful to give information about the context to the target word; furthermore, word ordering is taken into account, which was not the case for Word2Vec.

Another variant of Doc2Vec other than the already explained PV-DM is PV-DBOW (Paragraph Vector - Distributed Bag Of Words). While PV-DM is similar to the CBOW variant of Word2Vec, PV-DBOW is similar to the SkipGram variant of Word2Vec. In

PV-DBOW, the paragraph vector is given as input of the model ignoring the context words in the input and the task is to predict words randomly sampled from the paragraph, while PV-DM considers the concatenation of the paragraph vector with the word vectors to predict the next word in a text window. In addition to being conceptually simple, PV-DBOW requires storing less data. Both variants of Doc2Vec are represented in Figure 4.10; in Figure 4.11 can be seen the strong conceptual similarity between Doc2Vec and Word2Vec models.

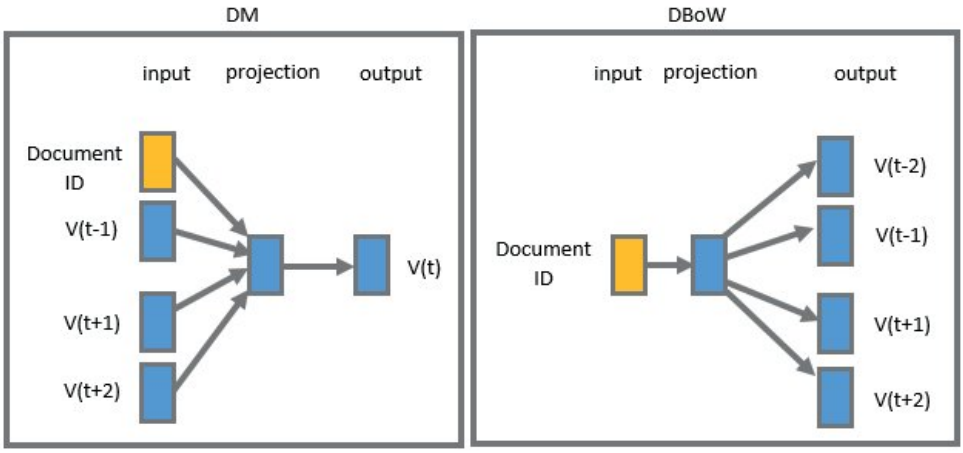


Figure 4.10: PV-DM and PV-DBOW variants of Doc2Vec. Picture taken from Bilgin et al. [4].

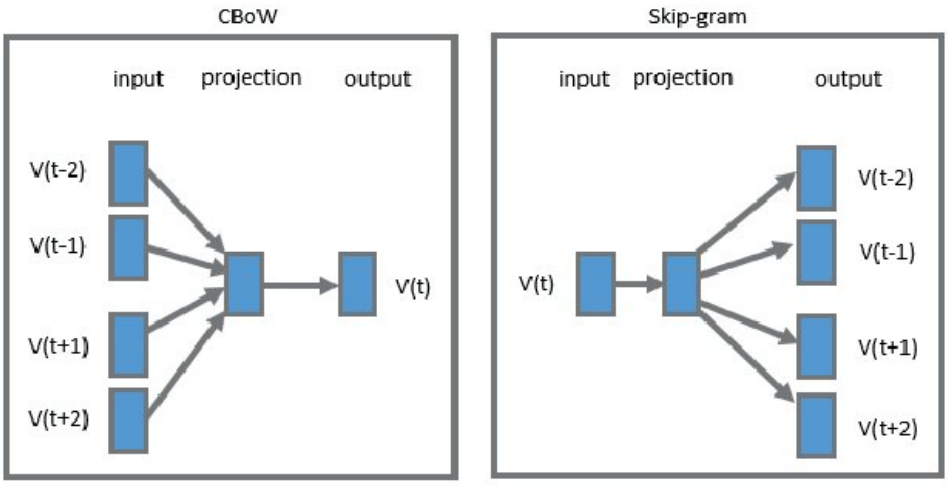


Figure 4.11: CBOW and Skip-Gram variants of Word2Vec. Picture taken from Bilgin et al. [4].

Doc2vec has proved to have a great performance on unsupervised tasks [7], for this reason, the Doc2Vec model is a good benchmark to see if a non-graph-based method outperforms

the InfoGraph model used in this thesis work. The results are presented in Subsection 5.1.6 following an unsupervised approach while in Subsection 5.1.8 Doc2Vec will be used to create embeddings of the documents and with a supervised approach, they will be evaluated.

4.4.3. Layer 1 documents labeling

A semi-automatic labeling has been produced using ChatGPT[8]. In particular, it has been asked to provide medical domain labels for each document, in a standard *json* format. Each document could be assigned multiple domains among [Orthopedics, Cardiology, Neurology, Endocrinology, Pulmonology, Nephrology, Oncology, Gastroenterology, Hematology, Infectious Diseases, Rheumatology, Psychiatry, Urology, Toxicology, Genetics, Pediatrics]. Therefore, some documents have been assigned to more than one label. The quality of these labels has been assessed by manual verification on half of the labeled documents.

The desired outcome of this thesis work is to have similar documents (in terms of medical meaning), belonging to the same cluster. To evaluate the outcome we decided to cluster the documents of layer 1 and 3 and using the semi-automatic domain labeling of the documents of layer 1 check if document embeddings produced by InfoGraph with the same domain labels ended up in the same cluster. The results of this approach are presented in Subsection 5.1.5. The same approach is then used to evaluate Doc2Vec embeddings in Subsection 5.1.6. A different approach based on a classic supervised set-up is presented in Subsection 5.1.8, where the labels are used for the training of the model, instead of the pure evaluation of the model only.

[8] `https://chatgpt.com/?oai-dm=1`

5 | Results

In this chapter, are presented the results of this thesis work, following the methodologies explained in the previous chapter. The discussion is divided into two sections:

- *Evaluation of the document embeddings and their cluster assignment*, where the output of the InfoGraph model i.e. the document embeddings, and their relative clusters are evaluated.
- *Hardware specification and computational time results*, where InfoGraph model, used for the E3C corpus, is analyzed from a computational point of view.

5.1. Evaluation of the document embeddings and their cluster assignment

In this section, we present the results of the embeddings produced by InfoGraph for the E3C dataset, analyzing the goodness of the embeddings themselves and of their clusterization results. To guide the reader through the different techniques, in Table 5.1 are presented all the methods employed for the evaluation of the results that will be discussed more deeply in the following subsections.

Task	Methodology
Role of the source and of the length of the documents	t-SNE visualization
Embeddings evaluation with UMLS	UMLS API, Kendall's Tau Correlation, HAC
Supervised approach to understand cluster assignments	k-means, Decision Tree Classifier
Clustering the documents divided by type of source	k-means, HAC
Embeddings evaluation with Doc2Vec	Doc2Vec
Clustering of InfoGraph embeddings on same length documents	k-means
Supervised approaches using ChatGPT labels	Decision Tree Classifier, TF-IDF, Doc2Vec

Table 5.1: Summary of the tasks and relative methods for clusters and embeddings evaluation. The methods are explained in detail throughout Section 5.1.

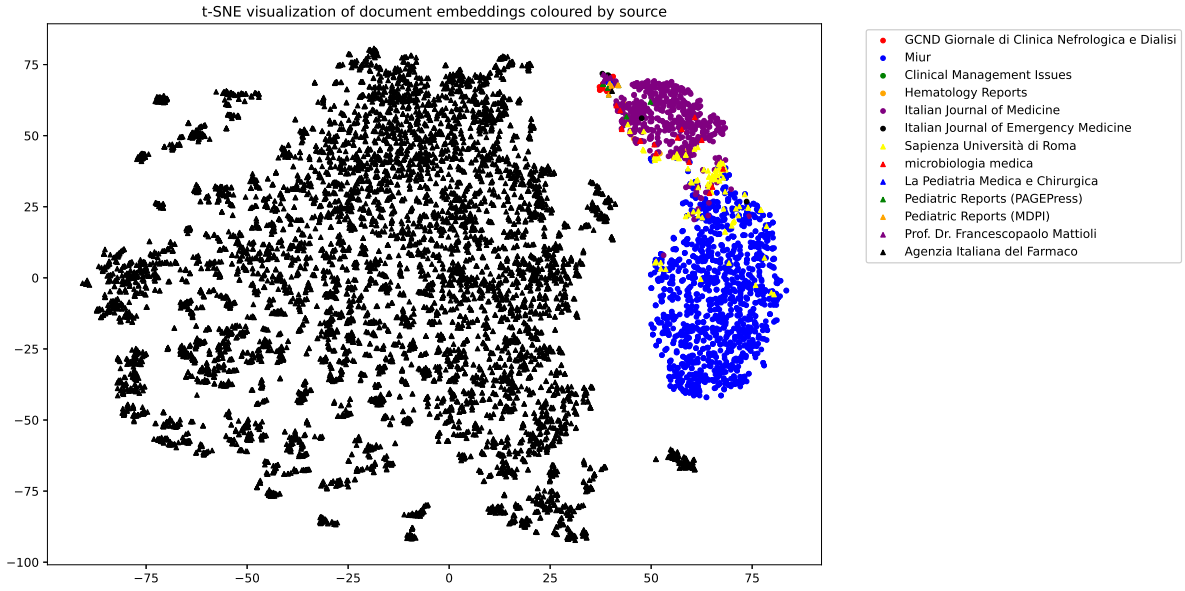


Figure 5.1: Visualization with t-SNE of the documents embeddings of layer 3 produced by InfoGraph, grouped by source. Documents coming from AIFA form a distinct cloud of points.

5.1.1. Role of the source of the documents

By working on layer 3, a visualization analysis with t-SNE led to the hypothesis that some outliers were present in the data, indeed, while a distinct cloud of points was dense in a specific region of the plot, some points lay far apart from this cloud. Analyzing the outliers, each of those points was the representation of documents coming from Agenzia Italiana del Farmaco (AIFA). The Image 5.1 shows a visual representation produced by t-SNE of the embeddings of the documents belonging to layer 3, grouped by source. With this visualization, it was clear that it was not only that the outliers were coming from AIFA but all the documents coming from this source were completely separated from the other documents. As presented in Section 3.4, there are 8084 out of 10213 documents coming from AIFA in layer 3. By simply reading some of the documents with AIFA as a source, it was clear that those documents represent medicine leaflets.

Inspired by this important finding, where the source of the documents played a fundamental role in analyzing the document embeddings, we decided to conduct an analysis of all the different sources that are present on the E3C dataset (not only AIFA), trying to understand whether this characteristic of each document is able to provide information about the document embeddings.

The presence of the documents with AIFA as a source, raised some questions for us, firstly

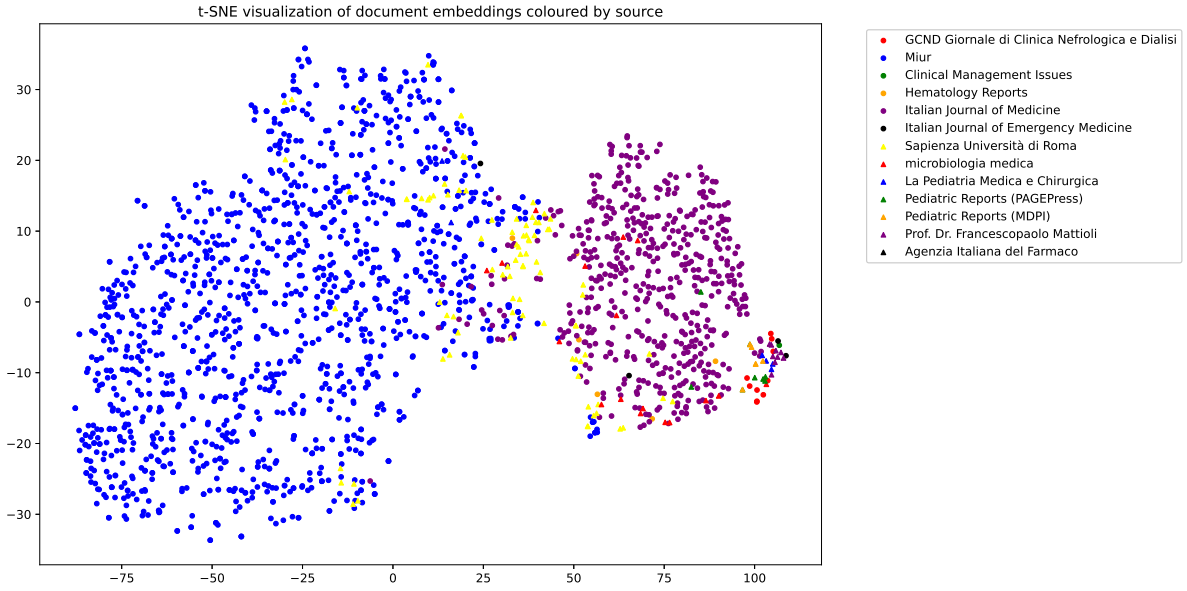


Figure 5.2: Visualization with t-SNE of the documents embeddings of layer 3 produced by InfoGraph (using also AIFA documents in training), grouped by source. The documents are highly separated according to their source.

whether to keep documents coming from AIFA or not for our clustering analysis, but since the goal of this thesis work is to cluster clinical case documents related to patients, we decided to discard them from the analysis since medicine leaflets don't generally provide an example of clinical information about a specific patient but rather it provides general information of the medicine, its usage, and its side effect. Figure 5.2 shows the visual representation using t-SNE of the document embeddings produced by InfoGraph using all the data at the disposal, i.e. using also the documents coming from AIFA. The t-SNE was produced using only documents which weren't from AIFA. In this setting, is clear that information on the source of the documents plays a fundamental role in embedding representation.

The second question was whether to use those documents as input for the embedding creation phase with InfoGraph and then clustering just the documents that contain clinical information; since medicine leaflets provide useful information about the drugs and about the diseases, a good option would be to run InfoGraph model without removing documents coming from AIFA, this presumably would lead to an enriched representation of all the other patient clinical case document embeddings, since InfoGraph is able to provide global information sharing across the different graphs (see again Figure 5.2 to see the visual result of this approach). On the other hand, since more than 80% of the documents of layer 3 come from AIFA, InfoGraph would presumably produce embeddings that are biased in

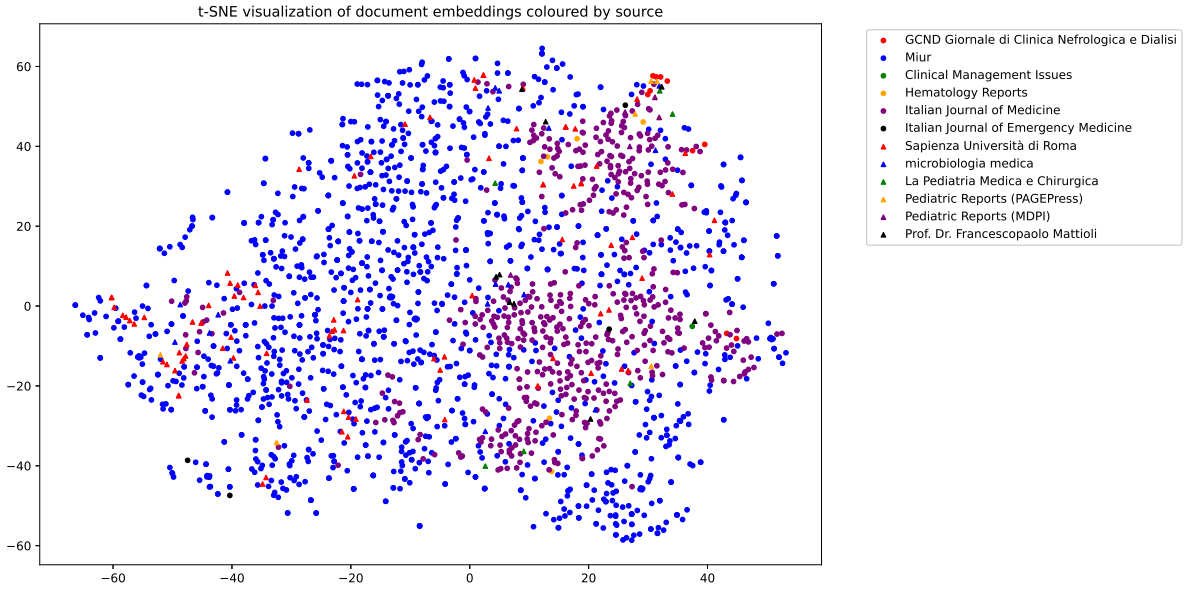


Figure 5.3: Visualization with t-SNE of the documents embeddings of layer 3 produced by InfoGraph (without using AIFA documents in training), grouped by source. The source of the documents slightly separates the documents.

the direction of medicine leaflets, which is not what we want as a final goal. For this last reason, we decided to run InfoGraph discarding all the documents coming from AIFA. To understand whether running InfoGraph on documents that don't have AIFA as a source, would lead to different results, we present in Figure 5.3 the t-SNE visualization of the document embeddings of layer 3, excluding the one coming from AIFA. The embeddings were produced by InfoGraph without giving AIFA documents as input. In this setting, the sources still play an important separating role, but less than in the previous experiments.

We note that our final goal is not to capture differences between documents coming from different sources but rather capture similarities and differences of the documents under a medical meaning framework, in order to find clusters of the E3C dataset that correspond to patients with similar disease, medical context and solutions to the clinical problem. For this reason, according to the previous experimental results, in our next analysis we will consider the embeddings produced by InfoGraph using only the documents with sources different from AIFA, since in this way, as previously mentioned, the sources influence with a minor intensity the produced embeddings.

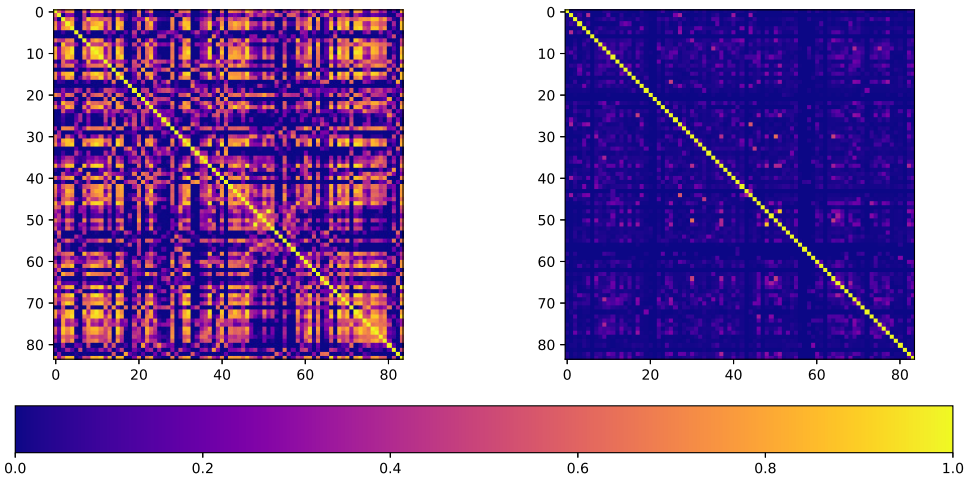


Figure 5.4: $SIM_{InfoGraph}$ on the left and SIM_{UMLS} on the right.

5.1.2. UMLS

As explained in Subsection 3.3, layer 1 comes with clinical entities annotation in the format of UMLS CUI. In this subsection, we present the evaluation of InfoGraph embeddings exploiting the information carried by the clinical annotations. We remind that layer 3 is not provided with clinical entities annotation and that as discussed in the previous subsection, AIFA documents have been removed.

By means of UMLS API's method explained in Subsection 4.4.1, we were able to create a similarity matrix for the documents of layer 1, excluding two documents that didn't present any clinical annotation, so the matrix SIM_{UMLS} has dimension 84×84. Analogously, InfoGraph embeddings of layer 1 have been put in the format of a similarity matrix using cosine similarity, and the two documents without clinical entities were removed from the analysis so that also $SIM_{InfoGraph}$ has dimension 84×84. The two similarity matrices can be seen in Figure 5.4 from which is possible to understand that the two methods are giving two very different similarity matrices, at least in magnitude of each value. In any case, the important information carried by the similarity matrices is which documents are more similar than other documents following either InfoGraph or UMLS method, so the magnitude of a single entry in a s similarity matrix should be compared with the other entries of the same similarity matrix.

After having created $SIM_{InfoGraph}$ and SIM_{UMLS}, the comparison of the results with InfoGraph and with UMLS API, can be done following two approaches: rank-based sim-

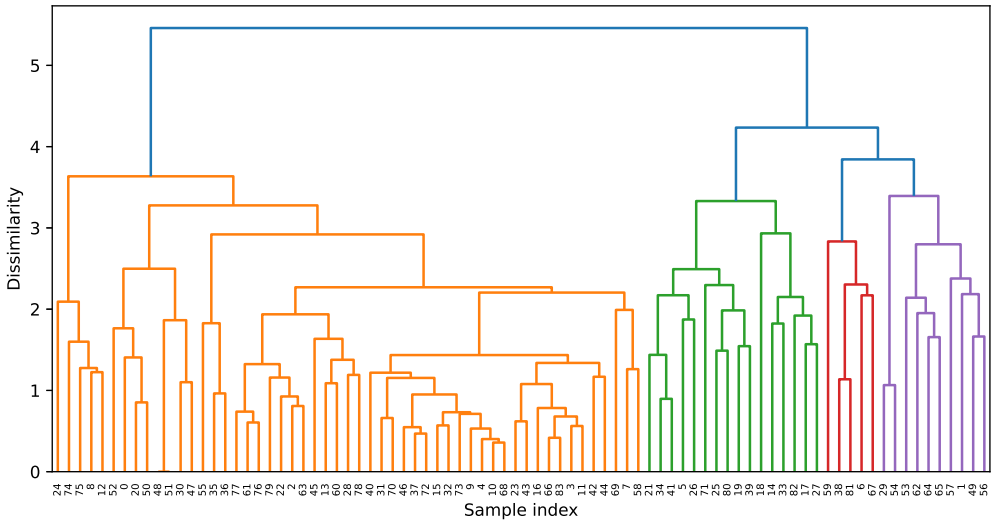


Figure 5.5: HAC dendrogram with average linkage of the SIM_{UMLS} similarity matrix.

ilarity and clustering.

We start by describing the rank-based similarity approach. Using the similarity matrices, for each target document is possible to rank every other document from the most similar to the least similar document compared to the target document. By doing so, a matrix 84×84 is created where for each row there is the rank of the most similar documents to the target one. This matrix is created both for the InfoGraph method and for the UMLS method. The two ranks for a fixed document (using InfoGraph and UMLS) are then compared using Kendall's Tau correlation index. This process is repeated for all the documents in layer 1. By averaging the results we get a mean correlation of 0.01672 with a p-value of 0.50244 so we can conclude that the correlation is not significant.

To follow a clustering approach instead, the Hierarchical Agglomerative Clustering (HAC) is preferred to k-means because the data are presented in the form of a similarity matrix so is impossible to compute distances between the points and the centroids for k-means, while with an agglomerative method, the similarity matrix is enough to compute the clusters. In Figure 5.5 is depicted the dendrogram of HAC with average linkage for SIM_{UMLS}. By directly reading the documents belonging to the same clusters according to HAC, it was clear that the results were not as good as expected since some different documents (in terms of medical meaning) were in the same cluster while some similar documents were not.

Summing up, both PyUMLSSimilarity and UMLS API methods could not evaluate In-

foGraph embeddings and clusters. The former method because of computational time issues, and the latter method because of the scarce robustness of the results. Both methods may become important in the future in similar scenarios if the previous issues are fixed. In the next subsections, we will discuss other methods to evaluate InfoGraph embeddings.

5.1.3. Role of the length of the documents

As presented in section 3.4, documents coming from different sources present many differences. The one that probably is more informative is about the length of the documents. AIFA documents are the longest one, while Miur and Sapienza Università di Roma are the shortest. As presented in the previous section, the source of the documents is a key characteristic indirectly learned by the embeddings created by InfoGraph. An important question that may arise is whether the length of the documents may be the real cause for the separation of documents based on their source, more than the source in itself. Indeed documents coming from the same source often have the same length; this is mainly because the publisher of the journal documents requires a fixed length for each production to be suitable for a publication. Documents that are not coming from journal publications are also subject to a similar process, for example, documents coming from Miur and Sapienza Università di Roma are texts representing clinical cases presented for exams of students in Medicine, for this reason, they always have a similar length.

We explained in the previous subsection that our goal is to find important insights on medical similarities within the clinical case documents and that using the sources as a proxy doesn't go in this direction. For the same thought process, using the length of the documents as a proxy to cluster documents is not in the scope of this thesis work.

We observe that in order to distinguish documents in "long" and "short" documents we used a threshold of 800 words: documents with less than 800 words are labeled as "short", and documents with more than 800 words are labeled as "long". To understand whether the length of the documents is providing different information from the one provided by the source of the documents, we performed a comparison of the visual representations performed by t-SNE of the produced document embeddings grouped by source and grouped by length. The two graphs show a similar pattern, long documents and short documents are generally separated in the graph in a similar way to the documents divided by sources, as shown in Figure 5.6.

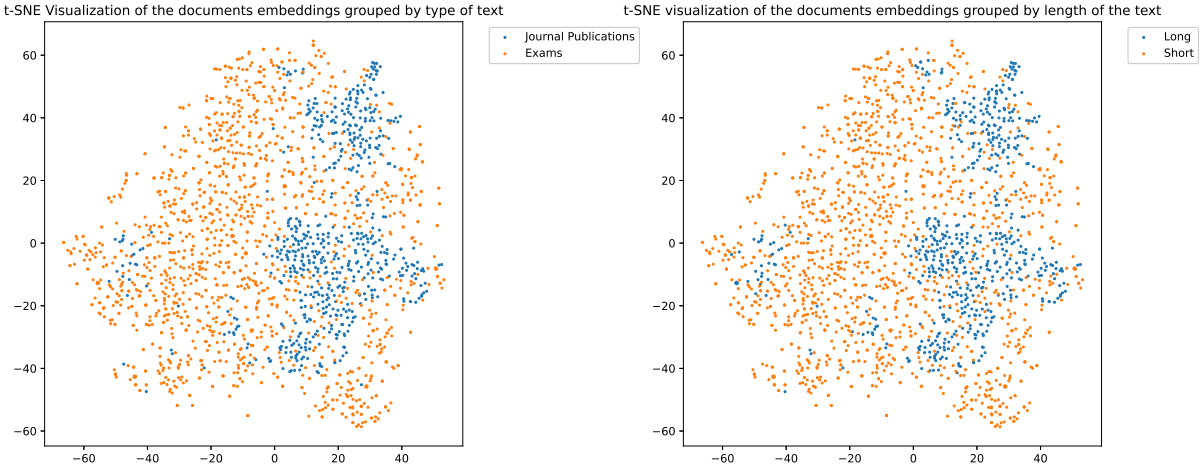


Figure 5.6: Visualization with t-SNE of the documents embeddings of layer 3 produced by InfoGraph (without using AIFA documents in training), on the left grouped by type of source (Miur and Sapienza Università di Roma are referred to Exams and the other sources to Journal Publications), on the right grouped by length category (using 800 words as separating threshold). The two graphs are almost identical, with some dense agglomeration of points belonging to the same category.

5.1.4. Supervised approach to understand the role of length and source of the documents

To prove experimentally that the information about the length of the document and its source, carried the same separating power for the document embeddings, we performed a k-means clustering algorithm to cluster the embeddings based solely on their vector representation produced by InfoGraph, with $k = 2, 3$. The idea of this clustering approach is to divide the embeddings into categories and then create a classifier model that based on the information of the documents (i.e. their length and their source) is able to predict the cluster assignment produced by k-means. If those two variables are able to produce good results in terms of the classification of the documents, it would mean that those variables are important for distinguishing the document embeddings from each other.

Going further with the details, after performing the k-means clustering [40] on the document embeddings produced by InfoGraph, we performed a decision tree classifier [53] with the length and source of the documents as input to predict the clustering assignment. The results showed an accuracy of around 60% for $k = 2$ and around 50% for $k = 3$, showing that those specific information of the documents are able to partly identify the clusters. Removing one of the two features (length and source of the documents), the results of the decision tree classifier are almost the same, meaning that there is a high correlation

between those two features in terms of information carried to the decision tree model. The results are shown in Table 5.2, and Table 5.3.

Value of K	All features	Drop length feature	Drop source feature
2	0.551	0.582	0.542
3	0.441	0.455	0.427

Table 5.2: Accuracies of the decision tree classifier model by changing the value of K of the k-means and choosing different sets of features between "source" and "length".

Value of K	All features	Drop length feature	Drop source feature
2	0.550	0.535	0.540
3	0.417	0.378	0.402

Table 5.3: Recall of the decision tree classifier model by changing the value of K of the k-means and choosing different sets of features between "source" and "length".

An approach that has been tested in order to reduce the effect of the length of the documents on the embeddings, is to change the READOUT function used in InfoGraph for E3C. The default choice implemented by the authors for the READOUT function is summation, as shown in Equation 4.7. In the case of nonnegative real-valued embeddings, the summation as READOUT function would lead to a bias given by the length of the documents, which is reflected in the graph number of nodes, because the embeddings relative to big graphs would be created by a summation of many nonnegative numbers, leading to a high result for the final graph embeddings. In our case, the word embeddings produced by CBOW that are given to InfoGraph as input are not nonnegative but both positive and negative real numbers, in any case, to test whether the length of the documents would create a bias for the document embeddings, we ran InfoGraph changing the READOUT function from *summation* to *mean*. The generated embeddings with this variation of the model didn't produce any change concerning the aforementioned analysis. For this reason, we discarded the choice of changing the default READOUT function for the InfoGraph model. In conclusion, some bigger structural changes, other than just changing the READOUT function, are needed to let InfoGraph learn medical information about the patient and not learn hidden structures present in documents coming from the same source and with the same length.

After having discussed in the previous paragraphs, about the information that the length of the documents and their sources is reflected in the document embeddings, and about the strong relation between the length and the source of a document, we decided to divide our

input document data into two groups: the documents coming from Miur and Sapienza Università di Roma, which refer to documents used for medical students examination, create one group of data in which we will work and all the documents coming from the other sources, which refer to documents of journal publications, will be the second group of data for our further steps.

The above approach of separating the documents is done with the idea of keeping away information about the sources and about the lengths of the documents. In this way, documents in the two groups are similar in terms of source, where they are linked for the type of source (journal or exam text), and, as a consequence are similar also in length. Removing this critical source of variability would possibly enhance the medical information carried by the document embeddings produced by InfoGraph.

To evaluate whether the separation of the documents into two groups (based on the type of source) leads to a weaker dependency between similar embeddings of length and source information of the document, we repeated the same method as before, where we produced with k-means a clusterization of the points and trained a decision tree model with length and source of the documents as input features to classify the documents against the ground-truth labels produced by k-means.

The accuracy and recall results for the documents belonging to exam sources are similar to the ones given by the documents without separation, so no improvements are encountered for exam documents. For documents belonging to journal publication sources instead, an unexpected increase in accuracy and recall is encountered. Those results led us to the conclusion that this separating method even if it was promising would probably not help our final clustering analysis. In the next section, we will present the results of this approach.

5.1.5. Clustering the documents divided by type of source

As discussed in the previous section, dividing the documents by type of source (journal publication or exam text) still causes a bias based on the length of the documents. Nevertheless, we present the results and the approach used in order to understand whether the embeddings produced by InfoGraph are able to capture medical meaning.

The embeddings of the documents are produced by InfoGraph using all layer 1 and layer 3 (excluding AIFA documents) with no distinction about the source of the documents, and retrospectively they are separated according to the source type.

The desired outcome of this analysis and in general of this thesis work is to have similar

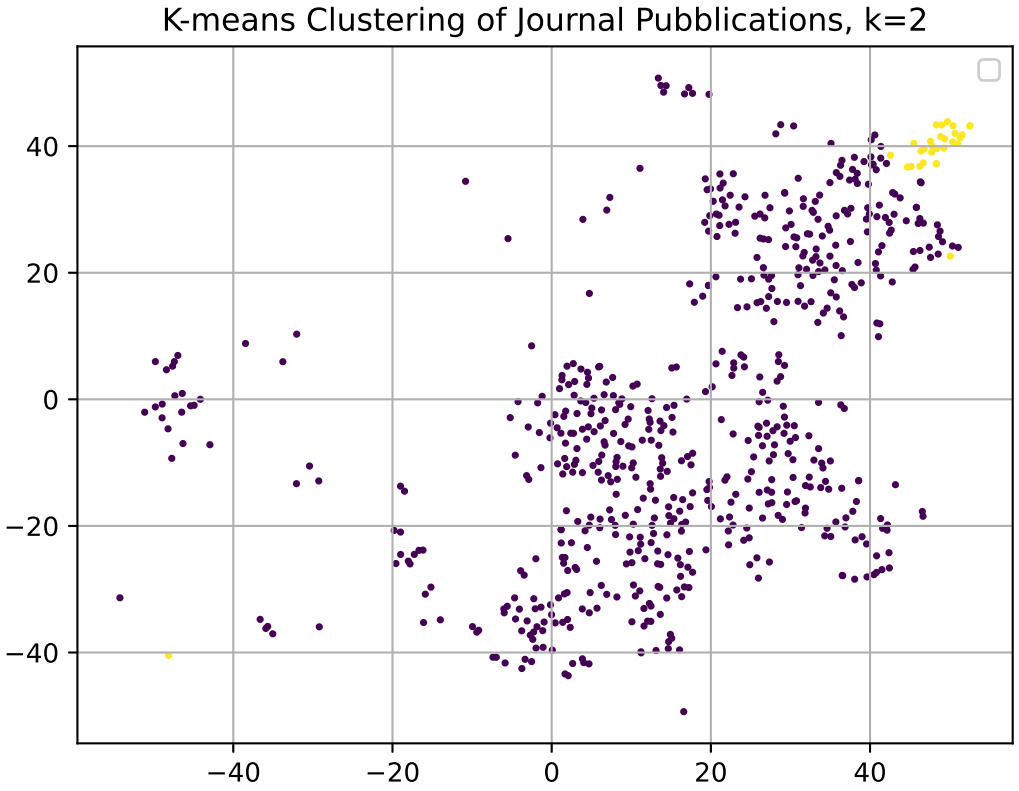


Figure 5.7: k-means clustering of the document embeddings with journal as the type of source.

documents (in terms of medical meaning), belonging to the same cluster. To evaluate the outcome we decided to cluster the documents of layer 1 and 3 divided by source type and using semi-automatic domain labeling of the documents of layer 1 check if documents with the same domain labels ended up in the same cluster.

We start the analysis with clustering performed on the document embeddings with journal publications as the type of source. Silhouette scores for k-means by changing k suggest k = 2 as the best choice. The dendrogram produced by hierarchical agglomerative clustering suggests 3 clusters instead. A t-SNE visualization of the clusters produced by k-means and hierarchical agglomerative clustering is depicted in Figure 5.7 and in Figure 5.8 respectively.

For both clustering approaches, we want now to evaluate the results in terms of the capability of capturing medical information. As explained before, for each document belonging to layer 1 we have labels representing the medical domain to which the text

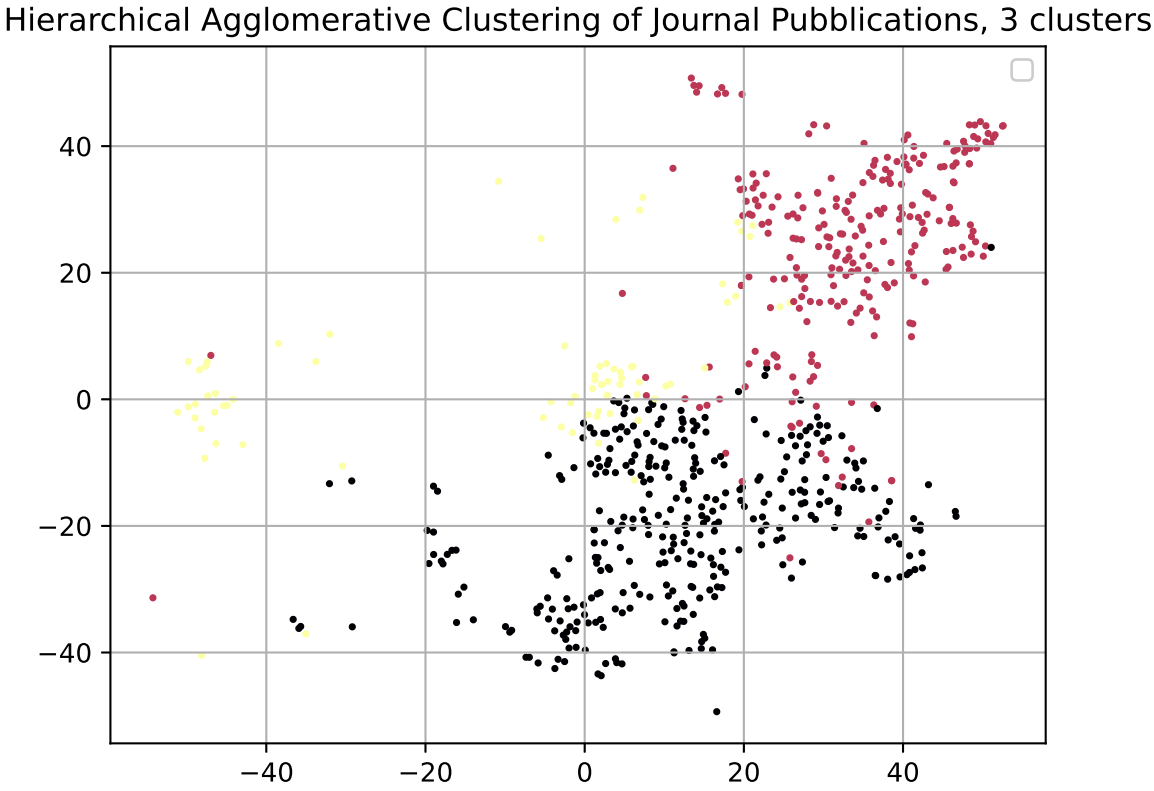


Figure 5.8: Hierarchical agglomerative clustering of the document embeddings with journal as the type of source.

is referred. We firstly evaluate visually if the documents with the same labels are close to each other, and secondly we see how many documents of the same domain are in which cluster (Figure 5.9). The desired result is to have documents with the same labels belonging to the same cluster (Table 5.4 and Table 5.8).

The domain labels given to each document can be very specific, but each document is provided with multiple labels so both specific and high-level domains are considered. For our analysis, we considered for simplicity just the 4 most frequent domain labels: *Cardiology*, *Pediatrics*, *Gastroenterology*, *Oncology*.

The t-SNE visualization of the document embeddings with the information of the domain of each document is depicted in Figure 5.9.

The assignment of each document embeddings to the clusters created by k-means and hierarchical agglomerative clustering are shown in Table 5.4, Table 5.5, Table 5.6, Table 5.7 for the k-means and in Table 5.8, Table 5.9, Table 5.10, Table 5.11 for the hierarchical

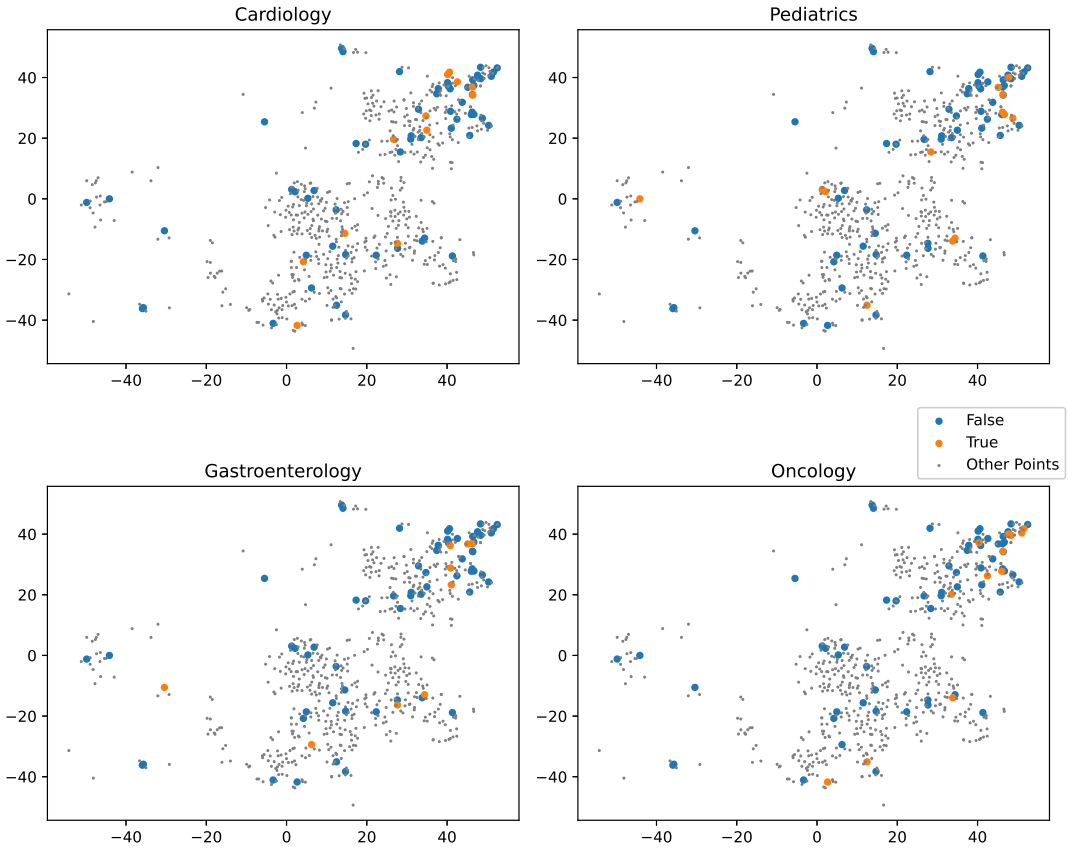


Figure 5.9: The points assigned as True are the document embeddings belonging to layer 1 with the domain label specified in the title of the sub-plot noted as True; the points assigned as False are the ones with the domain label noted as False. Other Points are the document embeddings of layer 3 which do not have domain labels. The documents represented are only the subset of documents belonging to journal publication sources.

agglomerative clustering.

	Cluster 1	**Cluster 2**
False (Layer1)	49	10
True (Layer1)	10	2
Layer 3	583	18

Table 5.4: Number of documents belonging to each cluster, divided by Cardiology domain label. Clusters created by k-means clustering.

Both with a visual analysis (Figure 5.9) and by counting the documents with the same domain in the clusters with the tables, we proved that the embeddings produced by

	Cluster 1	Cluster 2
False (Layer1)	48	10
True (Layer1)	11	2
Layer 3	583	18

Table 5.5: Number of documents belonging to each cluster, divided by Pediatrics domain label. Clusters created by k-means clustering.

	Cluster 1	Cluster 2
False (Layer1)	52	7
True (Layer1)	10	2
Layer 3	583	18

Table 5.6: Number of documents belonging to each cluster, divided by Gastroenterology domain label. Clusters created by k-means clustering.

	Cluster 1	Cluster 2
False (Layer1)	49	10
True (Layer1)	10	2
Layer 3	583	18

Table 5.7: Number of documents belonging to each cluster, divided by Oncology domain label. Clusters created by k-means clustering.

	Cluster 1	Cluster 2	Cluster 3
False (Layer1)	16	35	8
True (Layer1)	4	8	0
Layer 3	325	212	64

Table 5.8: Number of documents belonging to each cluster, divided by Cardiology domain label. Clusters created by hierarchical agglomerative clustering.

InfoGraph did not capture the medical meaning, furthermore splitting the data by source doesn't improve the results. We observe that the previous analysis, conducted on journal publications documents, has been applied also to exam documents, but layer 1 has too

	Cluster 1	Cluster 2	Cluster 3
False (Layer1)	17	36	5
True (Layer1)	3	7	3
Layer 3	325	212	64

Table 5.9: Number of documents belonging to each cluster, divided by Pediatrics domain label. Clusters created by hierarchical agglomerative clustering.

	Cluster 1	Cluster 2	Cluster 3
False (Layer1)	17	38	7
True (Layer1)	3	5	1
Layer 3	325	212	64

Table 5.10: Number of documents belonging to each cluster, divided by Gastroenterology domain label. Clusters created by hierarchical agglomerative clustering.

	Cluster 1	Cluster 2	Cluster 3
False (Layer1)	17	34	8
True (Layer1)	3	9	0
Layer 3	325	212	64

Table 5.11: Number of documents belonging to each cluster, divided by Oncology domain label. Clusters created by hierarchical agglomerative clustering.

few documents with this type of source, so the results are not statistically significant, in any case, similar results are expected also for those documents.

In the next sections, we will show the different methods applied trying to overcome the issue of the bias produced by the length of the documents.

5.1.6. Doc2Vec

To evaluate doc2vec embeddings of E3C corpus dataset, we repeated the main analysis done for the document embeddings produced by InfoGraph, as presented in the previous sections. Doc2vec embeddings were produced with GENSIM python library [56][1], using

[1] `https://radimrehurek.com/gensim/`

40 epochs and vector size 50.

Figure 5.10 shows the t-SNE projections of the document embeddings divided by source. There is a clear pattern of documents belonging to the same source lying close to each other in the image. The same result occurred for document embeddings produced by InfoGraph, as shown in Figure 5.3. This first visual analysis suggests that the information about the source of the documents is a strong proxy for the embeddings.

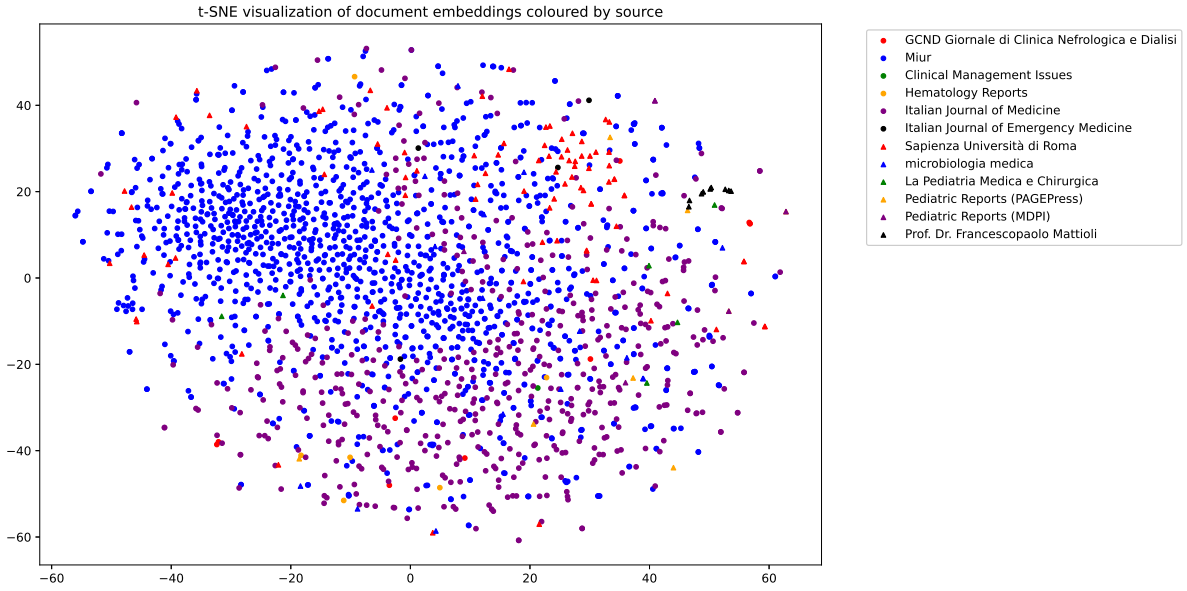


Figure 5.10: Visualization with t-SNE of the documents embeddings of layer 3 produced by the doc2vec model (without using AIFA documents in training), grouped by source. The source of the documents slightly separates the documents.

In order to evaluate if doc2vec is capable of capturing the medical meaning of a text, we used the labels of layer 1 documents produced by ChatGPT to visually understand by means of t-SNE if documents with the same domain label are similar to each other (i.e. near to each other in the t-SNE projections). Figure 5.11 shows the results of this analysis which proves that the medical information has not been captured by doc2vec; Table 5.12, Table 5.13, Table 5.14, Table 5.15, shows the distributions of documents embedding created by doc2vec, assigned to each cluster computed by hierarchical agglomerative clustering. Once again, comparing this picture with the counterpart produced by InfoGraph document embeddings and the relative tables, as shown in figure 5.9, we can conclude that both methods gave similar outcomes.

In conclusion, while doc2vec proved to perform well on many textual datasets, on the E3C dataset was not able to capture significantly the medical meaning of the documents but it was able to capture relations depending on the different sources of the documents.

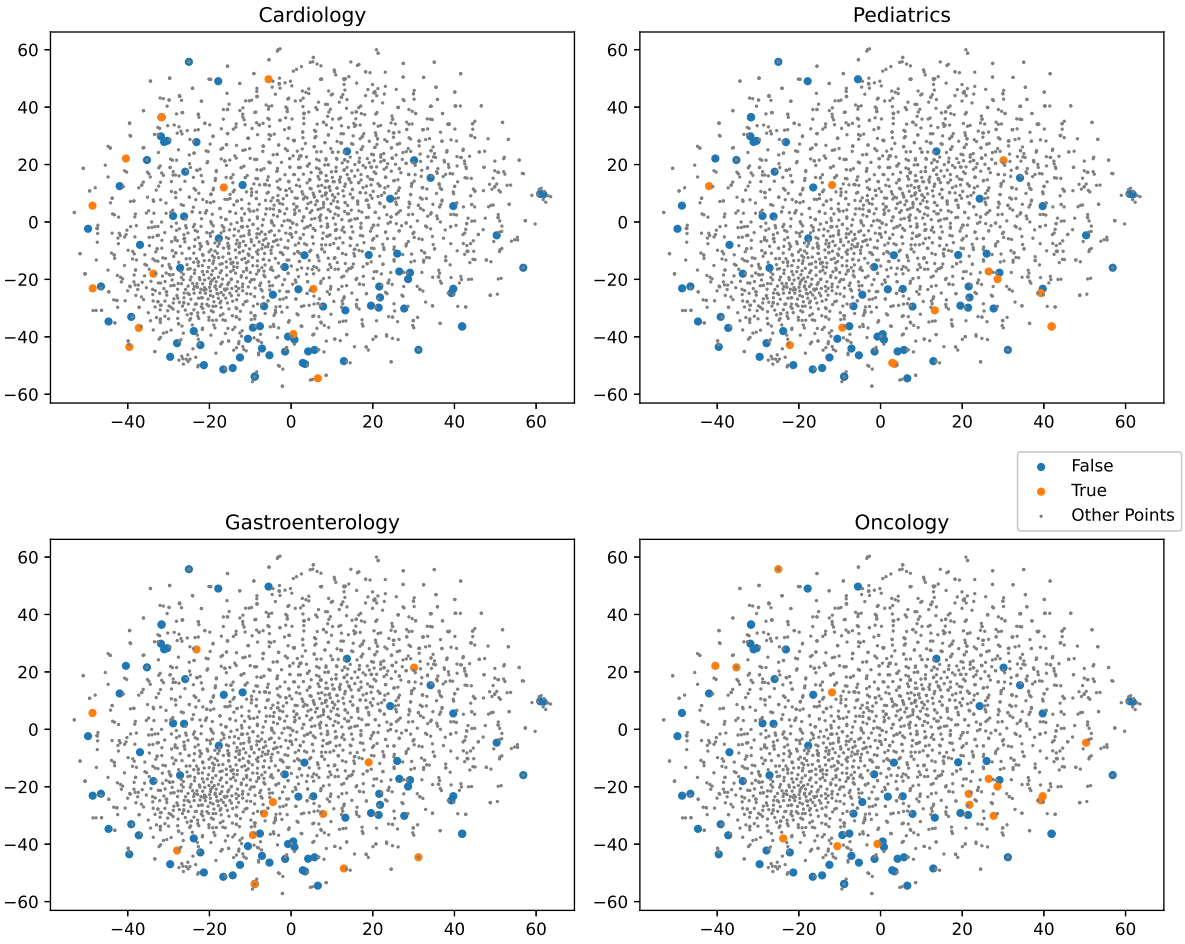


Figure 5.11: The points assigned as True are the document embeddings belonging to layer 1 with the domain label specified in the title of the sub-plot noted as True; the points assigned as False are the ones with the domain label noted as False. Other Points are the document embeddings of layer 3 which do not have domain labels.

The two aforementioned observations are in common with the document embeddings produced by InfoGraph. One reason for this result may be the quality of the textual data provided by the E3C dataset or the fact that hidden information within the texts have been considered more important by both unsupervised methods.

5.1.7. InfoGraph on same length documents

As explained in the previous sections, the length of the documents (and as a consequence their sources) appeared to be a strong bias for the document embeddings production done by InfoGraph. The reason for this lies in the learning structure of InfoGraph, which is able to learn local and global similarity between graphs, and for instance is able to

	Cluster 1	Cluster 2	Cluster 3
False (Layer1)	17	49	8
True (Layer1)	5	7	0
Layer 3	323	199	64

Table 5.12: Number of documents belonging to each cluster, divided by Cardiology domain label. Clusters created by hierarchical agglomerative clustering.

	Cluster 1	Cluster 2	Cluster 3
False (Layer1)	20	45	8
True (Layer1)	2	11	0
Layer 3	323	199	64

Table 5.13: Number of documents belonging to each cluster, divided by Pediatrics domain label. Clusters created by hierarchical agglomerative clustering.

	Cluster 1	Cluster 2	Cluster 3
False (Layer1)	20	48	6
True (Layer1)	2	8	2
Layer 3	323	199	64

Table 5.14: Number of documents belonging to each cluster, divided by Gastroenterology domain label. Clusters created by hierarchical agglomerative clustering.

	Cluster 1	Cluster 2	Cluster 3
False (Layer1)	20	44	7
True (Layer1)	2	12	1
Layer 3	323	199	64

Table 5.15: Number of documents belonging to each cluster, divided by Oncology domain label. Clusters created by hierarchical agglomerative clustering.

distinguish graphs with different structures and the length is an important characteristic of the structure of a graph. We observe that TUDatasets[46], i.e. the graph datasets that are used to evaluate InfoGraph on graph classification tasks, have input graphs with

similar size.

An important characteristic of InfoGraph is its possibility to transform variable-sized inputs into fixed-sized outputs, this strategy has been applied for example by Seki et al. [61]. In this section, we tried to go against this useful ability of InfoGraph hoping to enhance the results for our specific problem, but is important to notice once again that InfoGraph in itself doesn't need to have inputs of the same size to work properly.

To go in the direction of learning medical information about the documents we attempted a strategy that will not be accomplished for reasons that will be explained in the following paragraphs. We started transforming the documents of E3C into documents with the same length according to a simple rule: documents with more than a fixed threshold T of words will be cut to T words and documents with less than T words will be repeated (and in case cut). We observe that while the statistics on the length of the documents consider each word of the data, to transform the documents into a fixed-size dataset, we applied the transformation to the preprocessed input texts, meaning that, as explained in 4.1 much fewer words are present since stop-words have been already removed. For this reason, the choice of the threshold T to which uniform the length of the documents has been chosen to be the mean of the length of the documents without considering AIFA documents, i.e. $T = 52$ words. Following this rule we transformed every document into a document with a fixed size of T.

We observe that this approach was an extreme solution in order to account for medical information on the documents and is not a perfect solution, indeed some information on documents longer than T words will be lost, and on the other hand the information of documents shorter than T will be redundant, but in this way, we are sure that there is no more bias given by the length of the documents.

The problem with this approach is that when creating the adjacency matrix for the graphs representing the documents, short documents are repeated to increase their length up to T, but for the way we constructed graphs out of text, each node is a different word in the text, so repeating the text, would not lead to an increase of the dimension of short documents graphs. With this in mind, the approach cannot be applied.

To overcome the above issue but still go in the same direction of working on same length documents in order to neutralize the bias created by the length of the documents, we decided to work on separate sources, since as explained in the previous sections, the source of the documents and its length share the same trend.

In particular, to study the performance of exam-type documents, we will consider only

documents coming from MIUR documents, and to study the performance of journal publication documents, we will consider only documents coming from the Italian Journal of Medicine. The choice for these two sources out of the other ones is just because of their numerosity, indeed the other sources provide fewer documents, and training InfoGraph only with those documents is problematic. In this way, we can study the embeddings of each type of source separately from all the others reducing the noise created by having text written with different syntax and form.

A k-means clustering approach has been conducted on the two separate corpora, by changing $k \in \{2, 3, 4, 5, 6, 7, 8\}$. The results, even with this attempt, didn't show any relevant pattern or clustering able to reflect the medical domain of the documents of layer 1, which has been used as before for the evaluation. In the following lines, are shown four random examples of documents belonging to the same cluster, coming from Miur. There is no same pattern of the meaning between those examples.

Example 1:

Un uomo di 48 anni al termine di un viaggio in corriera da Parigi a Roubaix per una strada tradizionale ad acciotolato, all'arrivo in albergo, manifesta dolore a sede lombo-iliaca sinistra, molto intenso, continuo con parossismi, irradiato anteriormente e medialmente fino allo scroto ed alla radice della coscia di sinistra con eliminazione di poca urina molto densa.

The translation of the previous Italian text is:

A 48-year-old man at the end of a coach trip from Paris to Roubaix on a traditional cobbled road, on arrival at his hotel, manifests pain at the left lumbil-iliac site, very intense, continuous with paroxysms, radiating anteriorly and medially to the scrotum and the root of the left thigh with elimination of very little urine very thickly.

Example 2:

In un paziente ricoverato per politrauma in terapia intensiva, dopo 10 giorni di terapia con antibiotici a largo spettro si sviluppa una grave infezione polmonare sostenuta da un batterio Gram negativo resistente a quasi tutte le sostanze ad attivit antibatterica.

The translation of the previous Italian text is:

In a patient hospitalized for polytrauma in the intensive care unit, a severe pulmonary infection sustained by a Gram-negative bacterium resistant to almost all substances with antibacterial activity developed after 10 days of therapy with broad-spectrum antibiotics.

Example 3: *Una donna di 20 anni accusa da alcune settimane una diarrea muco-ematica*

e dolori addominali. Le biopsie del colon-retto mostrano infiammazione cronica diffusa al retto ed al sigma, microascessi criptici ed alterazioni strutturali delle cripte.

The translation of the previous Italian text is:

A 20-year-old woman has been complaining of mucohematous diarrhea and abdominal pain for several weeks. Colorectal biopsies show chronic diffuse inflammation in the rectum and sigma, cryptic microabscesses, and structural changes in the crypts.

Example 4:

Una donna all'ottavo mese di gravidanza ha presentato nelle ultime settimane un discreto aumento di peso dovuto in parte ad una ritenzione di liquidi: Durante la notte si lamenta di dolore alla mano destra con "punture di spilli" al palmo e alle prime due-tre dita della mano specie dal lato palmare.

The translation of the previous Italian text is:

A woman in the eighth month of pregnancy has presented a fair amount of weight gain in recent weeks due in part to fluid retention: During the night she complains of pain in her right hand with "pinpricks" in the palm and the first two to three fingers of the hand especially on the palmar side.

5.1.8. Supervised approaches using layer 1 labels

Since no relevant results came up proving the ability of InfoGraph to capture the medical meaning of the documents of E3C, we decided to test the E3C dataset in a classical supervised fashion. As explained in 4.4.3, documents of layer 1 have been labeled by ChatGPT according to the medical domain of the documents, allowing us to build supervised machine learning models and understand their performance.

Each document can have multiple labels but in order to follow an easier supervised approach, for each input only a single label has been retained: if many labels were provided, the most frequent label across layer 1, has been assigned. Furthermore, to diminish the noise given by having many labels, only the 4 most frequent domain labels are considered, while the remaining labels are grouped under a unique label called "Other".

The first step, before testing the E3C dataset with supervised classification models, is to transform the text into a mathematical format, as we have done with InfoGraph. We started with TF-IDF which is an easy but effective method to represent texts. Anyway, there is some important information to keep in mind, indeed layer 1 only has only 86 documents so the number of features in the input should be very low. TF-IDF pro-

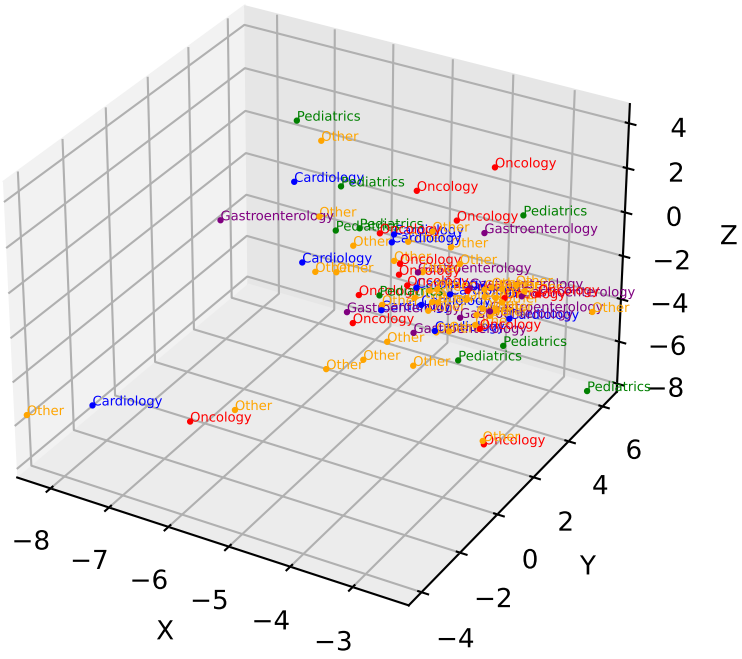


Figure 5.12: 3D visualization of Doc2Vec embeddings for the documents of layer 1, colored by the domain label of each document produced by ChatGPT.

duced document embeddings with 3850 size, and therefore it was problematic to proceed. Following a PCA dimensionality reduction, the first components retained a very low percentage of explained variability therefore the dimensionality problem of TF-IDF cannot be fixed and another approach has been considered. Following the aforementioned needs, we decided to rely on Doc2Vec, which allows to directly choose the size of the embeddings. In order to evaluate the results visually the dimension of the Doc2Vec embeddings was chosen to be 3. In Figure 5.12 is shown the representation of the 86 document embeddings plotted in the 3D computed by Doc2Vec. No relevant pattern can be observed from the plot.

We proceeded to perform a decision tree classifier on the data, dividing the 86 documents into train and test sets, encoding the target labels as one-hot encoding. The results of precision, recall, and f1-score, evaluated on micro-average, macro-average, weighted average, and sample average showed poor results comparable to random guesses. The results a re shown in Table 5.16. In conclusion, even with a supervised method, for the

E3C dataset was tough to capture the different medical domains of each document. This result can encourage InfoGraph embeddings since it is possible that with other datasets the results can be better.

Class	Precision	Recall	F1-Score	Support
Cardiology	0.25	0.33	0.29	3
Gastroenterology	0.00	0.00	0.00	2
Oncology	0.33	0.17	0.22	6
Other	0.33	0.44	0.38	9
Pediatrics	0.25	0.17	0.20	6
micro avg	0.27	0.27	0.27	26
macro avg	0.23	0.22	0.22	26
weighted avg	0.28	0.27	0.26	26
samples avg	0.27	0.27	0.27	26

Table 5.16: Classification report with precision, recall, F1-score, and support for each class. The results refer to the test set.

5.2. Hardware specification and computational time results

We ran InfoGraph with a normal laptop without GPU but just with CPU (11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz, 32,0 GB ram), this confirms that even if the learning structure of InfoGraph requires many computations (for every node in every graph in the batch mutual information is computed against all the graphs in the batch) it can be considered an accessible model. Highlighting its ability to run effectively on standard hardware configurations, such as a normal laptop, without the need for specialized cloud computing or GPU resources. Making InfoGraph a viable option for researchers and practitioners who may not have access to high-performance computing infrastructure.

The default hyperparameters used for InfoGraph are Learning Rate = 0.001, Hidden Dimension = 16, Number of Graph Convolutional Layers = 3, and Number of epochs = 10. In the following paragraphs, we want to show how different choices of the hyperparameters affect the computational time. The portion of the E3C corpus considered for the analysis are layer 1 and 3 documents, discarding AIFA documents. This leads to much faster results given the numerosity of AIFA documents and their length.

In Figure 5.13 is shown the execution time with respect to the Hidden Dimension. In-

creasing the Hidden Dimension reflects a linear increase in the execution time. We notice that the Hidden Dimension size is an important parameter of the model since it is capable of providing more information on each node. Therefore, by increasing the Hidden Dimension size, the loss function has a steeper decrease, meaning that with fewer epochs the model reaches the optimal solution. We used Hidden Dimension size 16 as default to have a trade-off. In Figure 5.14, is well represented the aforementioned observation, where the loss of InfoGraph model associated with smaller Hidden Dimension Size takes more time to reach a stable solution.

In Figure 5.15 is shown the execution time with respect to the number of Graph Convolutional Layers. Increasing the number of Graph Convolutional Layers, the execution time increases but following a nonlinear trend. A similar consideration to the one provided for the Hidden Dimension size is necessary: increasing the number of Graph Convolutional Layers, the receptive field of each node is increased so even if more time is required, the information from nodes that are close to each other is better captured, leading to a faster decrease of the loss function.

By changing the learning rate instead, no changes in the execution time are detected since the computations derived from this change are not affected in terms of time.

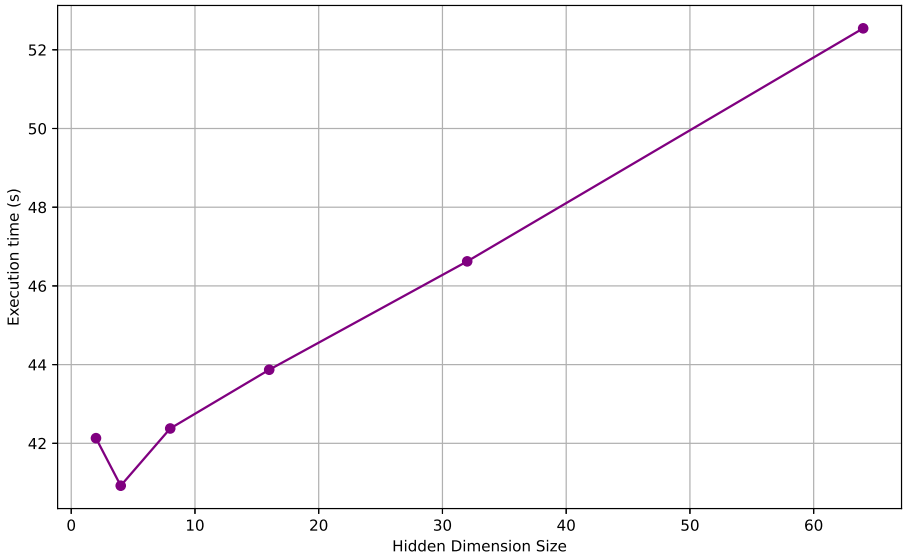


Figure 5.13: Execution time in seconds varying the Hidden Dimension Size of InfoGraph node embeddings.

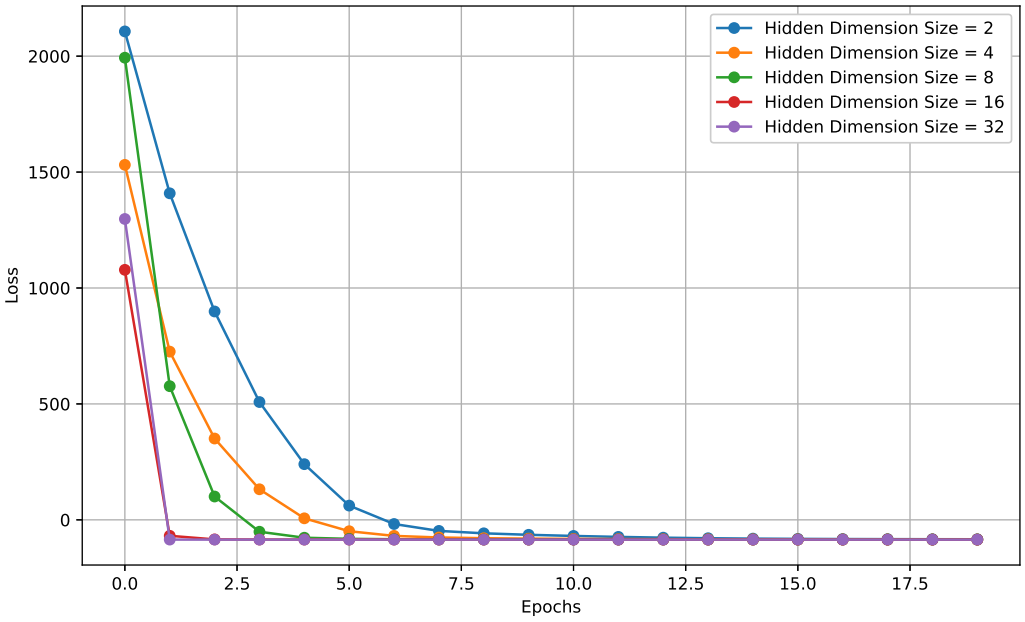


Figure 5.14: Losses values at each epoch, by changing the Hidden Dimension Size of the InfoGraph model.

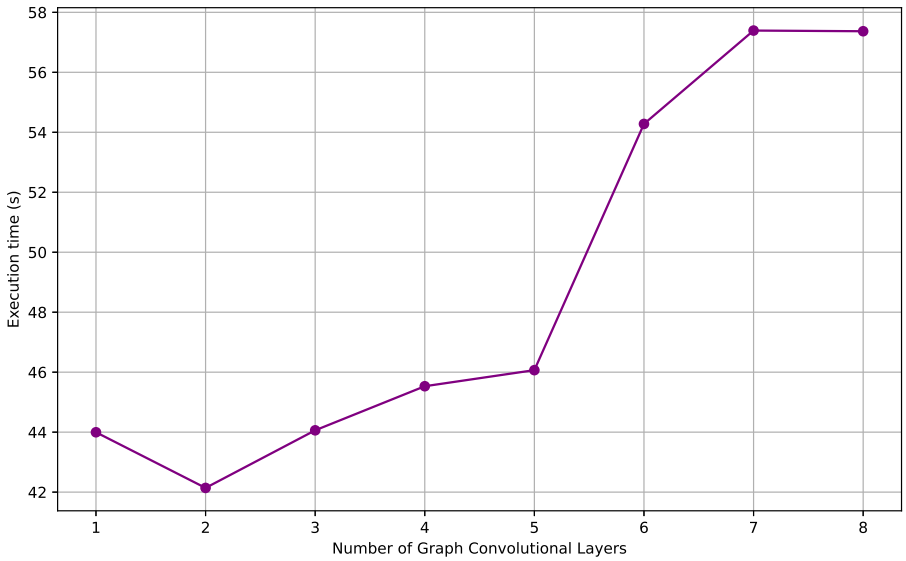


Figure 5.15: Execution time in seconds varying the number of Graph Convolutional Layers of InfoGraph model.

6 | Discussion and Conclusions

6.1. Discussion

Inspired by the fact that approaches based on Graph Neural Networks achieved state of the art results in supervised text classification, in this thesis work we analyzed the outcomes of GNNs on pure absence of labels for text clustering. The texts have been preprocessed and then translated into graphs where each node was associated with a word in the text and the edges were associated with the link between adjacent words. Using the self-supervised InfoGraph model we produced embeddings for each document belonging to the E3C corpus. In the next paragraphs, we present the main results obtained in the course of our analysis.

Summarizing the results of Chapter 5 we can conclude that the InfoGraph model, applied to the documents of the E3C corpus, was able to create a valuable representation of the text in the form of document embedding. Nevertheless, the document embeddings were capturing information regarding the source and the length of the documents, more than the medical meaning associated with the text. By removing the AIFA documents that are not the focus of this thesis work and then rerunning the model, the document embeddings started to show a less marked distinction between documents from different sources. Still, the sources appeared to play an important role. In particular, documents coming from sources like Sapienza Università di Roma and Miur, which are texts related to medical exams submitted to university students, appeared to be similar to each other in terms of embeddings, and analogously, journal publications coming from sources like the Italian Journal of Medicine had similar embeddings. In order to reduce the noise provided by the source type (exam documents and journal publications), we decided to split the corpus into two corpora, one for each source type, so that the information carried by the medical meaning can be isolated and therefore enhanced.

Comparing the results of the similarity matrix created by InfoGraph embeddings and the one created by exploiting the CUIs annotation in layer 1, no medical information appeared to be detected. Nevertheless, the information extracted from our UMLS methods was

limited and this didn't provide a reliable approach to evaluate the results. To have that UMLS can be used as a benchmark for medical meaning in the E3C corpus, it needs to provide more information about the broader concepts related to a CUI.

The next method used to evaluate the results is Doc2Vec. Generally, Doc2Vec is able to capture the meaning of the documents within a corpus but with our specific case of the E3C dataset, the results appeared to be similar to InfoGraph embeddings, where the medical meaning was not captured. The ChatGPT medical domain labels of layer 1 confirmed these results.

The last method used to evaluate InfoGraph embeddings was a pure supervised approach, where the Doc2Vec embeddings of layer 1 were trained and tested using a decision tree classifier in order to predict the medical field label provided by ChatGPT. The results border with random guesses. In this last trial, where a model like Doc2Vec which has already proved to work effectively in similar scenarios, didn't provide good results, raised the idea that the problem was the E3C dataset, rather than the model used to find medical meaning within the documents. In the following section, we dive into similar questions and provide possible future developments that can be deployed in the next researches.

From the computational point of view, with our choice of the graph to represent the documents belonging to the E3C, InfoGraph proved to be a valuable solution compared to other similar approaches like BERT and transformer-based models, where the needed computational time for the training is often onerous.

6.2. Future developments

In the literature, very few researches focus on GNNs for unsupervised learning and none uses GNNs for document clustering, to the best of our knowledge. For this reason, many improvements and tests can be implemented subsequent to our analysis. The key directions in which we think that future efforts should focus are:

- Test different GNN architecture.
- Enhance the graph structure to represent the text.
- Test on other datasets.

In the evaluation part of InfoGraph, we mainly focused on comparing the results with other models, but in doing so we hardly relied on the correctness of the other models which cannot be taken for granted. Another development that could help the evaluation of the results for E3C is UMLS. Indeed UMLS official website provides valuable information

regarding broader and narrower concepts of a CUI; the same does not hold for UMLS API, where less information is returned. By enhancing UMLS API, using clinical entities as pseudo labeling of the documents could provide improvements in the results. Having labels also allows a supervised or semi-supervised learning setting but this would not go in the direction of our purpose i.e. to build a model capable of cluster documents without labels.

The second key improvement that can be implemented is the choice of the construction of the input graph. For this thesis work, we relied on a simple graph where the nodes are the words of the text and the edges between the words are the link between adjacent words. Following a generalization schema, as presented in TextLevelGCN, is possible to increase the parameter p i.e. the window size, in order to connect words more distant from each other. Furthermore, other types of graphs can be given as input to the InfoGraph model, for example, creating syntactical or semantical edges between words rather than solely proximity edges as done in our analysis. Another option is to create heterogeneous graphs, where nodes can be words, concepts, sentences, or documents. Lastly, is possible to change the focus from graph clustering to node clustering to still tackle the problem of document clustering, in the sense that in this thesis work, each graph represented a document while the problem can be translated into a setting where one big graph represents the entire corpus and the nodes represent the documents; the edges between nodes can be for example the similarity between documents, information about the source or about the authors. Changing the graph structure could enhance the ability of the model, which can be for instance, InfoGraph or other self-supervised GNN models, to capture the information regarding the medical domain.

The last improvement that can be pursued is to find a way to capture the medical meaning of a text. This is indeed the main goal of this thesis work and has not been accomplished. Starting from the first steps that we performed to translate the text into document embeddings, the choice of Word2Vec as word embeddings for our analysis is questionable. Indeed Word2Vec embeddings cannot capture the context in which the words are appearing and they are static, so the same word which in two sentences is referring to something different, with Word2Vec embeddings it takes exactly the same value. Training the InfoGraph model with dynamics embeddings should improve the capability of capturing the medical meaning of the documents. Another issue with word embeddings is that some medical terms appear very rarely so the trained embeddings are not representative of the real word, since the E3C corpus does not have enough documents. For this reason, a preprocessing phase of removing or fixing uncommon and unfrequent words can be further investigated. The next approach in order to let the model better grasp the

medical meaning of the documents is to change the input graph structure, as previously discussed. Lastly, a consideration of the E3C corpus has to be made; the documents belonging to E3C come from different sources and therefore the model is biased toward capturing the hidden differences between different sources, such as the different lengths of the documents, and the form in which the documents are written. In our analysis, we split the corpus according to the source type trying to reduce this bias but in doing so we also reduced a lot the numerosity of our corpus. In future trials is possible to take into consideration changing the dataset, and using a more specific corpus, with documents similar in structure but different in terms of medical meaning. In this way, the model can focus on learning the medical meaning of the documents rather than be distracted from learning the noise produced by having documents with different structures.

6.3. Conclusion

In conclusion, a pipeline for clustering clinical case documents based on graph neural networks was developed in this thesis work. The pipeline was applied to a medical dataset of Italian texts (E3C). The method is a first attempt to exploit GNNS for document clustering; it has potential and captures some text features. However, in this specific application, it did not provide the expected results, which should have been more related to the clinical meaning of the documents.

Bibliography

[1] I. Alonso and D. Contreras. Evaluation of semantic similarity metrics applied to the automatic retrieval of medical documents: An umls approach. Expert Systems with Applications, 44:386–399, 2016.

[2] E. Alsentzer, J. R. Murphy, W. Boag, W.-H. Weng, D. Jin, T. Naumann, and M. McDermott. Publicly available clinical bert embeddings. arXiv preprint arXiv:1904.03323, 2019.

[3] F. M. Bianchi, D. Grattarola, and C. Alippi. Spectral clustering with graph neural networks for graph pooling. In H. D. III and A. Singh, editors, Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pages 874–883. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/bianchi20a.html`.

[4] M. Bilgin and İ. F. Şentürk. Sentiment analysis on twitter data with semi-supervised doc2vec. In 2017 international conference on computer science and engineering (UBMK), pages 661–666. Ieee, 2017.

[5] S. Bird, E. Loper, and E. Klein. Natural Language Processing with Python. O'Reilly Media Inc., 2009.

[6] O. Bodenreider. The unified medical language system (umls): integrating biomedical terminology. Nucleic acids research, 32(suppl_1):D267–D270, 2004.

[7] A. Budiarto, R. Rahutomo, H. N. Putra, T. W. Cenggoro, M. F. Kacamarga, and B. Pardamean. Unsupervised news topic modelling with doc2vec and spherical clustering. Procedia Computer Science, 179:40–46, 2021.

[8] M. Bugueño and G. de Melo. Connecting the dots: What graph-based text representations work best for text classification using graph neural networks? arXiv preprint arXiv:2305.14578, 2023.

[9] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk,

and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078, 2014.

[10] A. Conneau, G. Lample, M. Ranzato, L. Denoyer, and H. Jégou. Word translation without parallel data, 2018.

[11] M. R. Cowie, J. I. Blomster, L. H. Curtis, S. Duclaux, I. Ford, F. Fritz, S. Goldman, S. Janmohamed, J. Kreuzer, M. Leenay, et al. Electronic health records to facilitate clinical research. Clinical Research in Cardiology, 106:1–9, 2017.

[12] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. CoRR, abs/1810.04805, 2018. URL `http://arxiv.org/abs/1810.04805`.

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

[14] K. Ding, J. Wang, J. Li, D. Li, and H. Liu. Be more with less: Hypergraph attention networks for inductive text classification. arXiv preprint arXiv:2011.00387, 2020.

[15] V. P. Dwivedi and X. Bresson. A generalization of transformer networks to graphs. arXiv preprint arXiv:2012.09699, 2020.

[16] V. P. Dwivedi, C. K. Joshi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson. Benchmarking graph neural networks. Journal of Machine Learning Research, 24 (43):1–48, 2023.

[17] M. Fey and J. E. Lenssen. Fast graph representation learning with pytorch geometric, 2019.

[18] C. Friedman and N. Elhadad. Natural language processing in health care and biomedicine. Biomedical Informatics: Computer Applications in Health Care and Biomedicine, pages 255–284, 2014.

[19] M. A. Gianfrancesco, S. Tamang, J. Yazdany, and G. Schmajuk. Potential biases in machine learning algorithms using electronic health record data. JAMA internal medicine, 178(11):1544–1547, 2018.

[20] H. Hafidi, M. Ghogho, P. Ciblat, and A. Swami. Graphcl: Contrastive self-supervised learning of graph representations. arxiv 2020. arXiv preprint arXiv:2007.08025.

[21] W. L. Hamilton. Graph representation learning. Morgan & Claypool Publishers, 2020.

[22] S. Hassan, R. Mihalcea, and C. Banea. Random walk term weighting for improved text classification. International Journal of Semantic Computing, 1(04):421–439, 2007.

[23] K. Häyrinen, K. Saranto, and P. Nykänen. Definition, structure, content, use and impacts of electronic health records: a review of the research literature. International journal of medical informatics, 77(5):291–304, 2008.

[24] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. science, 313(5786):504–507, 2006.

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

[26] L. Huang, D. Ma, S. Li, X. Zhang, and H. WANG. Text level graph neural network for text classification, 2019.

[27] O. G. Iroju and J. O. Olaleke. A systematic review of natural language processing in healthcare. International Journal of Information Technology and Computer Science, 8:44–50, 2015.

[28] W. Jin, T. Derr, H. Liu, Y. Wang, S. Wang, Z. Liu, and J. Tang. Self-supervised learning on graphs: Deep insights and new direction. arXiv preprint arXiv:2006.10141, 2020.

[29] T. N. Kipf and M. Welling. Variational graph auto-encoders. arXiv preprint arXiv:1611.07308, 2016.

[30] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks, 2017.

[31] E. Kościałkowska-Okońska. Translating medical texts for legal purposes: A growing challenge for court translators and interpreters. Comparative Legilinguistics, 11:7–21, 2012.

[32] J. H. Lau and T. Baldwin. An empirical evaluation of doc2vec with practical insights into document embedding generation. arXiv preprint arXiv:1607.05368, 2016.

[33] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In International conference on machine learning, pages 1188–1196. PMLR, 2014.

[34] C. Leacock. Combining local context and wordnet similarity for word sense identification. WordNet: A Lexical Reference System and its Application, pages 265–283, 1998.

[35] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[36] Y. Li, P. Hu, Z. Liu, D. Peng, J. T. Zhou, and X. Peng. Contrastive clustering. In Proceedings of the AAAI conference on artificial intelligence, volume 35, pages 8547–8555, 2021.

[37] H. Linmei, T. Yang, C. Shi, H. Ji, and X. Li. Heterogeneous graph attention networks for semi-supervised short text classification. In Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP), pages 4821–4830, 2019.

[38] P. Liu, X. Qiu, and X. Huang. Recurrent neural network for text classification with multi-task learning. arXiv preprint arXiv:1605.05101, 2016.

[39] Y. Liu, M. Jin, S. Pan, C. Zhou, Y. Zheng, F. Xia, and S. Y. Philip. Graph self-supervised learning: A survey. IEEE Transactions on Knowledge and Data Engineering, 35(6):5879–5900, 2022.

[40] S. Lloyd. Least squares quantization in pcm. IEEE transactions on information theory, 28(2):129–137, 1982.

[41] B. Magnini, B. Altuna, A. Lavelli, M. Speranza, and R. Zanoli. The e3c project: European clinical case corpus. Language, 1(L2):L3, 2021.

[42] D. Marcheggiani and I. Titov. Encoding sentences with graph convolutional networks for semantic role labeling. arXiv preprint arXiv:1703.04826, 2017.

[43] A. T. McCray, O. Bodenreider, J. D. Malley, and A. C. Browne. Evaluating umls strings for natural language processing. In Proceedings of the AMIA Symposium, page 448. American Medical Informatics Association, 2001.

[44] B. T. McInnes, T. Pedersen, and S. V. Pakhomov. Umls-interface and umls-similarity: open source software for measuring paths and semantic similarity. In AMIA annual symposium proceedings, volume 2009, page 431. American Medical Informatics Association, 2009.

[45] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 26, 2013.

[46] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann. Tu-

dataset: A collection of benchmark datasets for learning with graphs. arXiv preprint arXiv:2007.08663, 2020.

[47] A. Névéol, H. Dalianis, S. Velupillai, G. Savova, and P. Zweigenbaum. Clinical natural language processing in languages other than english: opportunities and challenges. Journal of biomedical semantics, 9:1–13, 2018.

[48] S. Nowozin, B. Cseke, and R. Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. Advances in neural information processing systems, 29, 2016.

[49] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang. Adversarially regularized graph autoencoder for graph embedding. arXiv preprint arXiv:1802.04407, 2018.

[50] T. Pedersen, S. V. Pakhomov, S. Patwardhan, and C. G. Chute. Measures of semantic similarity and relatedness in the biomedical domain. Journal of biomedical informatics, 40(3):288–299, 2007.

[51] J. Pennington, R. Socher, and C. Manning. GloVe: Global vectors for word representation. In A. Moschitti, B. Pang, and W. Daelemans, editors, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar, Oct. 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL `https://aclanthology.org/D14-1162`.

[52] M. F. Porter. Snowball: A language for stemming algorithms, 2001.

[53] J. R. Quinlan. Induction of decision trees. Machine learning, 1:81–106, 1986.

[54] R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. IEEE transactions on systems, man, and cybernetics, 19 (1):17–30, 1989.

[55] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[56] R. Rehurek and P. Sojka. Gensim–python framework for vector space modelling. NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic, 3 (2), 2011.

[57] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics, 20:53–65, 1987.

[58] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. nature, 323(6088):533–536, 1986.

[59] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. Information processing & management, 24(5):513–523, 1988.

[60] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. IEEE transactions on neural networks, 20(1):61–80, 2008.

[61] T. Seki, Y. Kawazoe, and K. Ohe. Clinical feature vector generation using unsupervised graph representation learning from heterogeneous medical records. In AMIA Annual Symposium Proceedings, volume 2023, page 618. American Medical Informatics Association, 2023.

[62] S. S. Sonawane, P. A. Kulkarni, H. Balinsky, A. Balinsky, W. Jin, R. K. Srihari, F. Zhou, F. Zhang, F. Rousseau, and M. Vazigiannis. Graph based representation and analysis of text document: A survey of techniques. International Journal of Computer Applications, 96:1–8, 2014. URL `https://api.semanticscholar.org/CorpusID:3466557`.

[63] K. Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. Journal of documentation, 28(1):11–21, 1972.

[64] F.-Y. Sun, J. Hoffmann, V. Verma, and J. Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. arXiv preprint arXiv:1908.01000, 2019.

[65] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. Advances in neural information processing systems, 27, 2014.

[66] M. Tayefi, P. Ngo, T. Chomutare, H. Dalianis, E. Salvi, A. Budrionis, and F. Godtliebsen. Challenges and opportunities beyond structured data in analysis of electronic health records. Wiley Interdisciplinary Reviews: Computational Statistics, 13(6):e1549, 2021.

[67] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. Journal of machine learning research, 9(11), 2008.

[68] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

[69] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, et al. Graph attention networks. stat, 1050(20):10–48550, 2017.

[70] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm. Deep graph infomax. arXiv preprint arXiv:1809.10341, 2018.

[71] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang. Mgae: Marginalized graph autoencoder for graph clustering. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pages 889–898, 2017.

[72] B. Weisfeiler and A. Leman. The reduction of a graph to canonical form and the algebra which appears therein. nti, Series, 2(9):12–16, 1968.

[73] L. Wu, Y. Chen, K. Shen, X. Guo, H. Gao, S. Li, J. Pei, B. Long, et al. Graph neural networks for natural language processing: A survey. Foundations and Trends® in Machine Learning, 16(2):119–328, 2023.

[74] S. T. Wu, H. Liu, D. Li, C. Tao, M. A. Musen, C. G. Chute, and N. H. Shah. Unified medical language system term occurrences in clinical notes: a large-scale corpus analysis. Journal of the American Medical Informatics Association, 19(e1): e149–e156, 2012.

[75] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems, 32(1):4–24, 2020.

[76] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? arXiv preprint arXiv:1810.00826, 2018.

[77] S. Yan, Y. Xiong, and D. Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In Proceedings of the AAAI conference on artificial intelligence, volume 32, 2018.

[78] L. Yao, C. Mao, and Y. Luo. Graph convolutional networks for text classification. In Proceedings of the AAAI conference on artificial intelligence, volume 33, pages 7370–7377, 2019.

[79] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec. Hierarchical graph representation learning with differentiable pooling. Advances in neural information processing systems, 31, 2018.

[80] Y. You, T. Chen, Z. Wang, and Y. Shen. When does self-supervision help graph convolutional networks? In international conference on machine learning, pages 10871–10880. PMLR, 2020.

List of Figures

List of Tables

Ringraziamenti

Sono passati sei anni dal mio test di ingresso al Politecnico di Milano e mi trovo finalmente ora a tirare le somme su questo percorso. Molte persone sono rimaste nella mia vita da quel periodo. Voglio ringraziare i Bolzanesi che mi hanno fatto sempre sentire parte del gruppo nonostante vivessi in un'altra città, in particolare Tommaso, Stefano e Vittorio, che ciascuno a modo suo mi è stato vicino in questi anni. Gli amici del liceo, tra cui Edoardo che con dei semplici giri in bicicletta e chiacchierate profonde mi ha fatto spesso *staccare* dalla vita frenetica universitaria.

Voglio ringraziare la mia famiglia, mio papà per la sua razionalità e senso del dovere che mi ha passato durante tutta la vita e per il suo modo di fare il tifo per me in silenzio, con pochi *bravo*, ma sempre segnandosi tutti i miei esami su un quadernino per poter sapere quanto mi mancasse alla laurea. Ringrazio mia mamma per la sua dolcezza e sicurezza che mi ha sempre trasmesso, assieme al suo amore per la riflessione e per l'imparare, facendo invece lei il tifo per me ad alta voce. Ringrazio mio fratello per essere stato per me una guida e un esempio da seguire ed emulare, nell'università e non solo; lo ringrazio per i primi anni di università vissuti assieme che mi hanno fatto tornare a quando eravamo bambini e dormivamo in cameretta assieme. Ringrazio mia sorella per avermi dato sempre una buona motivazione per tornare a casa i weekend e passare del tempo con lei (assieme alla motivazione del ricevere le feste dalla Esme e dalla Sally). Ringrazio la nonna Franca e il nonno Giuliano per i sacrifici che hanno fatto per me e per gli altri cugini e per i valori che mi hanno trasmesso.

Voglio anche ringraziare le persone che mi hanno aiutato a essere ciò che sono, la maestra Clara alle elementari, la prof. Fornara alle medie, i prof del liceo che hanno creduto in me e i vari mister che ho avuto giocando a calcio.

Il Politecnico mi ha fatto crescere, facendomi capire il valore dell'impegno, della dedizione e soprattutto dell'umiltà, quindi mi sento di doverlo ringraziare. Ringrazio in particolare Vittorio che mi ha seguito costantemente durante questo lavoro di tesi, non facendomi sentire spaesato e insegnandomi molto. Ringrazio il Politecnico anche per gli amici che mi ha fatto conoscere durante questi anni come Jonny e Luca. In queste aule ho an-

che conosciuto Federica, prima in vesti di migliore amica e poi di fidanzata, e questo basterebbe per dover ringraziare il Politecnico.

Ringrazio Federica, per la condivisione delle esperienze universitarie e per essere stata un punto di riferimento per i primi anni di corso. Negli ultimi anni, per avermi ascoltato quando le raccontavo le mie giornate di studio, i miei esami, le mie idee per la tesi. La ringrazio per il tempo trascorso assieme, che non mi è mai abbastanza, e per avermi dato un obiettivo di vita.